

#### IV. APPENDIX

#### V. RELATED WORK

**Few-shot Imitation Learning.** Few-shot imitation learning is one of the main desiderata in robot learning, and more in general, in machine learning. Behaviour Cloning, where a neural network is trained over a dataset of observation action pairs coming from an expert demonstrator, often requires hundreds of demonstrations to learn tasks, making the goal of obtaining fast-learning robots laborious. Over the years, many methods have been proposed to tackle this problem. [11] proposed to train a network to explicitly take in a demonstration as input, and output actions that would follow the behaviour demonstrated. [23] proposed to learn a siamese network on pairs of demonstrations of the same behaviour to learn task-specific embeddings, able to then do one or few-shot learning at test time. [6] proposes to train several imperfect dynamics model to train a policy to perform efficient adaptation to real-world data. All these methods, however, still rely on gathering a substantial amount of interaction data. We propose to use the emergent pattern completion abilities of large Transformers pretrained solely on text, where no robotics data is used to pretrain or finetune it.

**Finetuning Large Pretrained Models** An effective paradigm that emerged from the recent research is to first pretrain large models on enormous, task-agnostic datasets, and then perform finetuning with substantially smaller datasets representing the task at hand. Such models are referred to as Foundation Models [18]. While being extremely successful in the fields of computer vision or natural language processing, the lack of vast robotics datasets hindered the creation of robotics specific Foundation Models. While [12, 7] demonstrated promising results by finetuning large Vision-Language Models on robotics data, learning new tasks is still inefficient and laborious. We demonstrate that is possible to effectively repurpose text-pretrained Transformers as general imitation learning machines without the need for any robotics data, and obtain results on par or superior to the current state-of-the-art.

**Use of Large Language Models in robotics as planners.** The exponential progress in capabilities of text-pretrained Transformers, or Large Language Models (LLMs), to interpret and reason over text generated a plethora of robotics works leveraging such models. [16, 25, 31, 21, 10] use LLMs as high-level planners, obtaining a list of steps to solve tasks in language form. [22, 34] leverage LLMs to map a high-level description of a task in natural language into rewards that can be used by a trajectory optimiser. While effective, these methods rely on LLMs to generate high-level plans or guidance for external optimisers, and therefore need additional techniques to ground these outputs into actions. In our work, we *do not use LLMs for natural language*, but instead exploit their pattern learning capabilities to directly process sequences of observations and generate executable low-level trajectories. We therefore shed light on a counter-intuitive phenomenon: *one of the most effective uses of LLMs in robot learning is achieved without the use of natural language neither in the*

*inputs nor in the outputs.*

**Language Models that output robotic actions.** [12] finetunes a Vision-Language Model with robotics data in the form of language annotated demonstrations. They obtain therefore a model that can receive as input a language instruction and output, similarly to our method, actions that can directly be executed by the robot. The main differences, however are that 1) we do not need finetuning, employing off-the-shelf models 2) while their model receives language instructions, our method can perform *in-context learning* on low-level demonstrations, therefore acting as a general imitation learning machine. [24] also demonstrated LLMs ability to output trajectories of executable actions instead of high-level textual plans. However, it can only generate and execute plans based on language instructions, therefore still relying on language as an input and being unable to learn from expert demonstrations. [26] was one of the first works to show the general pattern learning abilities of LLMs. They demonstrated the capability of LLMs to autoregressively complete a partial trajectory received as an input, therefore learning patterns beyond human language. However, their robotics exploration was not comprehensive, ignoring complex visual inputs. To the best of our knowledge, we are the first work to propose a complete framework that can learn from demonstrations in the forms of sequences of visual observations and actions.

**In-Context Learning** Scaling the training of large Transformers, especially on language data, led to the emergence of in-context learning [19]. While the phenomenon is not entirely clear, recent works have investigated its emergence in smaller, simpler to study and control settings [28]. [32] investigated the parallels between in-context learning, a purely feed-forward adaptation mechanism with no weights updates, and gradient descent, hypothesising that Transformers learn to implicitly apply a gradient update via attention heads. These observations inspired this work, and the use of Transformers and their in-context learning ability as general imitation learning machines. Notably, Transformers are not the only way to obtain in-context learning. [33] trained a graph neural network to learn objects alignments from few demonstrations. While they learn manipulation strategies via in-context learning like our method, they focus on finding relative poses between objects, while we can output end-effector actions, therefore resulting in more general behaviours, including non-prehensile manipulation.

#### A. Experimental Setup

We run our experiments using a Sawyer robot, interacting with objects on a table in front of it. The end-effector is a Robotiq 2F-85, on which we mount an Intel Realsense D435 RGBD camera. In every interaction episode the robots records one visual observation at the beginning of the task, then computes a trajectory of actions, and finally executes those. The camera is mounted on the wrist of the end-effector. The end-effector is moved 90cm above the table at the beginning of each episode to record an image observation.

During demo collection, the robot first captures an observation, and then the human expert maneuvers the end-effector

through kinesthetic teaching, while its  $SE(3)$  poses are recorded. Each demo therefore collects an RGBD observation and a sequence of end-effector poses. When collecting demos, the position of the objects for each task are randomised in order to cover part of the starting state distribution of the task. More details can be found in the Supplementary Material.

At test time, the objects are positioned on the table in novel configurations, designed to test both *interpolation* and *extrapolation* abilities of the tested method to the poses observed during training. The robot captures an observation as above, and then computes a series of end-effector poses.

We use **GPT-4 Turbo** [17] as Large Language Model, but later compare its performance to other models. In the Supplementary Material we provide more details about the way we give inputs to it and the way we process its outputs.

## B. Tasks

We thoroughly evaluate our method by choosing a family of everyday manipulation tasks and measuring its ability to efficiently replicate the expert behaviour, generalising after just few demonstrations. The criterion we adopted to choose the tasks is to 1) replicate tasks that appeared in the recent literature 2) measuring **KAT**'s ability to tackle a series of challenges that appear in everyday tasks. In particular, we chose tasks that collectively measure **generalisation to novel shapes, multi-modality, precision, dexterity, multi-stage execution, and 6D trajectories**. The tasks are the following:

- **Align T**: Inspired by [4], the robot needs to move a T-shaped object to align it to the axes of the table. The task requires *non-prehensile* abilities and the ability to model *multi-modal* distributions: demos are provided following different strategies (pushing the vertical or the horizontal part to align the T), and the robot needs to commit to a single one at test time.
- **Wiping a Plate**: The robot needs to follow the edge of a plate with a sponge in a circular motion. This task requires *multi-modal* reasoning: half the demos are given in clockwise direction, half are given counter-clockwise. The robot needs to commit to one at test time. We test for both interpolation and extrapolation abilities by also using unseen plates at test time of unseen dimensions.
- **Sweep**: The robot needs to sweep an object on a dustpan. The relative positions of dustpan and object are randomised, therefore the robot needs to compute an effective trajectory to bring the object on the dustpan. We test for generalisation by also using a novel dustpan and objects to sweep at test time.
- **Espresso**: The robot needs to insert an espresso capsule in a toy espresso machine, and then close its lid. This task requires noteworthy *precision*, and is composed of *multiple stages*.
- **Scooping**: The robot needs to scoop some chocolate powder from a cup with a spoon. The task was proposed in [35] as a challenging, *dexterous* task. We test for generalisation by using also unseen cups at test time.

- **Pick and Pour**: The robots needs to pick up a French Press and pour coffee into a cup. The pouring part of the task was proposed in [35] as a *challenging, dexterous* task. By also picking up the French Press, the task requires *multiple stages*. We test for generalisation by using also unseen cups at test time.
- **Hang**: The robot needs to hang a clothes hanger to an horizontal support. The task requires to reach a *precise position and height*, otherwise the hook will not hang to the support.
- **Put Bottle Upright**: The robot needs to pick up an horizontally placed bottle and place it upright on the table. This is also inspired from a task in [4], where a mug has to be flipped. The task requires a *non-trivial, 6D trajectory* to be completed, so that the bottle is stably placed. We test for generalisation by also using unseen bottles at test time.
- **Pick and Place**: The robot needs to pick up an apple and place it in a red bowl, with a blue bowl acting as a visual distractor. We test for generalisation at test time by both using an orange, and then a purple bowl as distractor instead of the blue bowl. This task requires *reasoning* abilities to correctly understand the goal given the demonstrations, and the ability to *ignore distractors*.

Additional details, like success criteria, are listed in the Supplementary Material.

## C. Baselines

As the goal of our work is to demonstrate the efficiency of Large Language Models (large text-pretrained Transformers) to effectively act as *imitation learning machines* by learning to emulate expert behaviour after few demonstrations, we compared our approach with **Diffusion Policies** [4], a state-of-the-art general imitation learning algorithm from the recent literature, that was already demonstrated to surpass a series of recent techniques in terms of learning efficiency and generality.

In particular, we first compare the vision-based Diffusion Policy method proposed in [4], and then modify it by first providing our proposed *keypoints* as input and *actions representation* as outputs (**KeyAct-DP**), therefore going from a (512, 512, 4) input space to  $K$  3D visual keypoints and from the original position and orientation representation of [4] to a triplet of 3D points uniquely representing an end-effector pose. We use the original authors' code, and optimise hyperparameters to maximise performance on our tasks.

These baselines allow us not only to compare **KAT** to the state-of-the-art in the field, but also to measure the contribution coming from the *keypoints* and *actions* representation we propose.

## D. Tasks Success Criteria

Here we describe the adopted criteria to define an episode as successful for each task.

- **Align T**: Given an imaginary frame of reference with axis  $x$  and  $y$  aligned with the longer and shorter sides of the

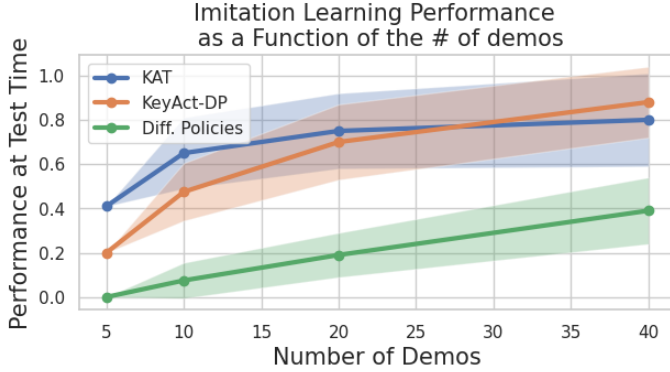


Fig. 5: Performance of each method as a function of the number of demos. While **KAT** outperforms the baselines in the few-shot regime ( $\leq 20$  demos), in-context learning struggles to improve as the number of demos increase even more. Plot shows mean and standard deviation across tasks.

table respectively, the long part of the T is withing 20 degrees to the  $y$  side.

- **Wiping a Plate:** The sponge grasped by the robot touched at least 80% of the edge of the plate.
- **Sweep:** The object to be swept is entirely on the dustpan at the end of the episode.
- **Espresso:** The espresso capsule is inserted in the slot and the lid is closed.
- **Scooping:** The robot scoops at least 5 grams of chocolate powder from the cup.
- **Pick and Pour:** The French press is inclined of at least 50 degrees from its normal vertical position while being held by the robot, with the spout pointing inside the mug. For practical reasons the French press was not filled with liquid, but we mimic a movement that would pour its contents into the mug.
- **Hang:** The cloth hanger its stably placed on its support.
- **Put Bottle Upright:** At the end of the episode, the bottle is in a stable vertical position when no longer grasped.
- **Pick and Place:** The apple (or orange) is entirely inside the red container.

#### E. Additional Experiments

**How does the performance change as the number of demonstrations increases?** In our previous experiments we provided 10 demonstrations for each task. To investigate the performance as a function of the number of demonstrations received, a key property in few-shot imitation learning, we chose a subset of the 9 tasks (*Align T*, *Wipe Plate*, *Espresso*, *Bottle Upright*) and provided 5, 10, 20 and 40 demonstrations for each task. We then compared the performance of KAT and the baselines. The results are plotted in Fig. 5. Diffusion Policy is unable to reliably learn a policy even with 20 demonstrations, as the input space is substantially higher-dimensional, but starts working when receiving 40 demonstrations, albeit with lower performance than the other methods. KeyAct-DP also obtains

	Clean	Distractors	Diff. BG	Both
<b>KAT (Ours)</b>	0.65	0.6	0.62	0.58

TABLE II: Effect of distractors and different background (BG) on the performance of our method.

remarkable results, but its performance evolves more slowly than KAT. However, when receiving 40 demonstrations, we can see its performance surpassing KAT on certain tasks. This suggests an important insight: pretrained Transformers can excel at in-context learning sequence-to-sequence patterns few-shot, making them particularly useful for low-data regimes, but currently do not scale well as further in-context data is provided. We hypothesise that with this amount of data leads in-context learning stops working optimally, and becomes a bottleneck. Instead, we believe that, with this amount of data, such Transformers may be finetuned to further improve performance, albeit the process is more laborious that in-context learning. Nevertheless, finetuning LLMs on Keypoint Action Tokens data is an interesting direction for future work.

**Can our method handle visual distractors?** To study robustness to distractors, we evaluate KAT on the *Align T*, *Wipe Plate*, *Espresso*, *Bottle Upright* tasks, providing 10 demonstrations per task, but at test time we add distractor objects, change the background, or both. For each of these scenarios, we run 10 test time episodes, reporting the results in Table III. These results showcase the robustness of KAT to visual distractors. This ability is obtained as a combination of two factors: from a vision perspective, the descriptors extracted from the observations via DINO-ViTs are robust to perturbations and have high dissimilarity across different objects. From a sequence-to-sequence pattern learning perspective, the pretrained Transformer is able to infer the *keypoints tokens* most relevant to the task by comparing the inputs-outputs relations in the demonstrations, therefore ignoring possible *keypoint tokens* that are less robust to unrelated visual changes.

#### F. Vision: Investigations on Keypoint Tokens

**What is the optimal number of keypoints to extract?** In this work we proposed a pipeline, based on the DINO family of Vision Foundation Models, to transform a visual observation  $o_i$  into a *sequence of 3D keypoints*  $k_{1:K}$ . The number of extracted keypoints  $K$  is an influential hyperparameters: a small  $K$  allows to strongly reduce input dimensionality, but generates a strong information bottleneck and may fail to capture more nuanced geometric and semantic information from visual observations. On the other hand, a large  $K$  allows to capture many visual details, but is then more susceptible to noise and visual distractors. We measured the performance of KAT on a subset of the 9 tasks (*Align T*, *Wipe Plate*, *Espresso*, *Pick and Place*) as a function of  $K$ , providing 10 demonstrations per task. We repeat the experimental procedure described in Sec. III-A 4 times, and extract  $K \in \{5, 10, 20, 40\}$  keypoints from the observations, varying it at each experiment. We measure the performance on 10 test time episodes, and report mean and standard deviation in the plot of Fig. 8. The



Fig. 6: The tasks we evaluated our method and the baselines on.

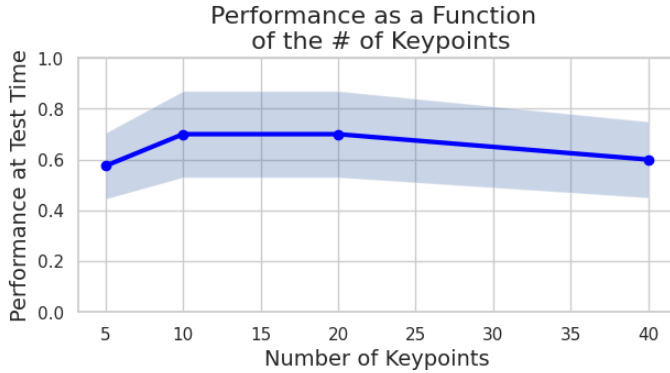


Fig. 7: Performance of our method as a function of the number of keypoints extracted. Plot shows mean and standard deviation across tasks.

results show that the optimal  $K$  lies between 10 and 20, with minimal variations in that range. We therefore chose 10 in our main experiments to reduce the computational complexity and input dimensionality.

**What is the best vision model to extract descriptors from?** To motivate our choice of using DINO-ViT as vision backbones of our work, we compared them to two other popular models in the robotics and computer vision literature: CLIP [29] and R3M [27]. We apply the same algorithm described in Fig. 3 with the difference that the dense descriptor tensors are extracted from CLIP or R3M (in the Supplementary Material we provide more information on this process). These experiments allowed us to test the robustness and representation abilities of these hidden representations. We run the same experimental setup described above in the previous subsection, setting  $K = 10$  and training and testing on *Align T*, *Wipe Plate*, *Espresso*, *Pick and Place*. Our results demonstrate that DINO is the optimal choice, coherently with the results from the recent literature [2]. While R3M is pretrained on data more related to robotics (first person videos of humans performing manipulation tasks [15]), interestingly it does not perform better than DINO, pretrained on a large task-agnostic dataset of web images [3]. This apparently counter-intuitive result was also reported in [8].

#### G. Action: Investigations on Action Tokens

**What is the optimal number of actions to record and predict?** In our work, actions are recorded as end-effector poses

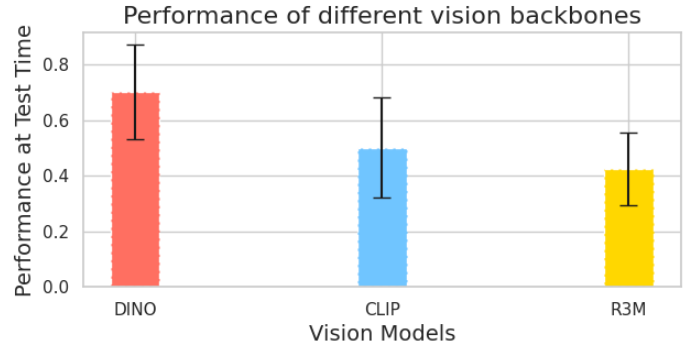


Fig. 8: Performance of our method when changing the vision backbone model from which we extract *keypoint tokens*. Plot shows mean and standard deviation across tasks.

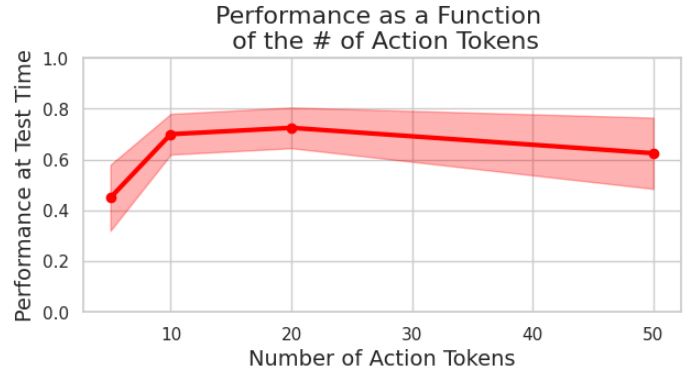


Fig. 9: Performance of our method as a function of the number of action tokens extracted from each demonstration trajectory, that strongly correlates with the number of action tokens generated at test time. Plot shows mean and standard deviation across tasks.

during demonstrations, then tokenised into *action tokens*, each being a triplet of 3D positions in the world frame. Vice versa, at test time the Transformer predicts a series of *action tokens* that are then transformed into end-effector poses and executed. An LLM Transformer is able to autonomously decide to stop the autoregressive generation of tokens, by generating an internal STOP token. The number of action tokens it predicts is often very close to the number of action tokens each demonstration was decomposed in.

Therefore, we investigate what is the optimal number of



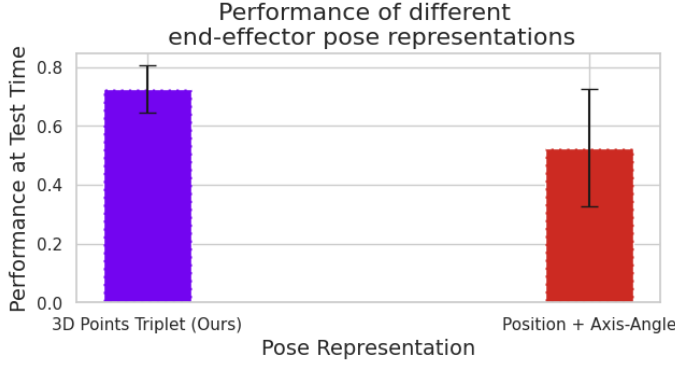


Fig. 10: Performance of our method with different representations of the end-effector poses. Plot shows mean and standard deviation across tasks.

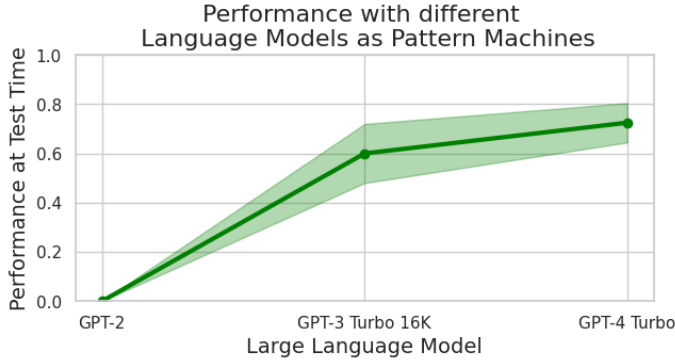


Fig. 11: Performance of KAT by varying the underlying LLM acting as imitation learning machine. Plot shows mean and standard deviation across tasks.

action tokens  $N_a$  to provide for each demonstration, to guide the test time generation. A small  $N_a$  reduces the length of the sequence and makes autoregressive generation easier and less prone to errors: however, it can also excessively subsample the trajectory of the end-effector, losing subtle but important movements. On the other hand, while a large  $N_a$  allows to precisely capture movements, it also makes both the input sequences excessively long and difficult to process, and also the generation phase prone to errors, as common to sequence-to-sequence networks tasked to predict very long outputs.

We replicate the experimental setup of Sec. V-F. We record 10 demos on the *Align T*, *Wipe Plate*, *Sweep*, *Bottle Upright* tasks at a very high frequency, and then subsample those to change the number of action tokens  $N_a$  each demonstration is decomposed into, with  $N_a \in \{5, 10, 20, 50\}$ . We then test the performance of KAT by providing to the Transformer these different quantities of tokenised actions extracted from the demonstrations, running 10 test time episodes per task.

Our results, in Fig. 9, show that the optimal number of tokens lies around 20: however, the drop at 50 is not large, suggesting the Transformer is still able to process this sequence length.

**What is the optimal representation of poses to use as action tokens?** In Sec. II-B we introduced the way we tokenise end-effector poses, transforming them through an invertible mapping into a triplet of 3D points  $e^w$ . Is this the most effective representation to represent, process and generate sequences of actions? To motivate our design choice, we compared our tokenisation pipeline to the representation of poses as 3D position and axis-angle representation for orientation. This representation is therefore composed of 6 numbers instead of 9, but the orientation part lies in a different, non-Euclidean manifold. As described in Sec. II-B, all representations share an additional number  $\delta_g$  representing the gripper open/closed state.

We replicate the above experimental setup, but compare the effect of representing and generating actions as action tokens  $e^w$  or as position and axis-angle representation. Results in Figure 10 demonstrate that our choice outperforms the more classical position/axis-angle method of representing  $SE(3)$  poses. As discussed in Sec. II-B, we hypothesise that representing poses as a set of Euclidean entities in the same frame as the keypoint tokens facilitates the reasoning process for Language Models, that instead struggle to generalise and interpolate/extrapolate correct axis-angle representations beyond the instances of the tasks observed during the demonstrations.

#### H. Free Robotics Lunch: Better Imitation Learning Machines by Scaling Language Models

Our work is inspired and designed around the finding that large Transformers pretrained on sequence-to-sequence tasks obtain the emergent capability to infer in few-shot patterns belonging to different data modalities. This effect has been observed especially in Large Language Models, large Transformers pretrained on vast amount of text. Scaling laws and empirical results have been obtained that allow the predict the capacity of these models as a function of the size of both their parameters and dataset.

As a result, the latest LLMs, like GPT-4 Turbo, can effectively act as few-shot imitation learning machines for robotics, reaching or surpassing the performance of state-of-the-art imitation techniques despite not needing any robotics data. As the field of LLMs has seen exponential growth in recent years [19, 14, 17, 13], we hypothesise that future LLMs will be even better (as suggested by numbers reported for the still unavailable at the time of writing Gemini Ultra [13]), more efficient imitation learning machines. This without the need for any particular innovation in robotics itself, be it in terms of data or algorithms. To concretely support this hypothesis, we evaluate and visualise the performance of different generations of LLMs.

We test KAT on the *Align T*, *Wipe Plate*, *Sweep*, *Bottle Upright* tasks, selecting  $K = 10$  and  $N_a = 20$ , but change the model of the LLM computing the output sequence given the demonstration sequences and the input sequence. We test three generations of GPT models: GPT-2 [1], GPT-3 Turbo 16k [19], GPT-4 Turbo [17]. Results in Fig. 11 demonstrate that the evolution of these text-pretrained Transformers, albeit

independent in any way from robotics, leads to an improvement in performance, as more recent models are more efficient at few-shot sequence-to-sequence pattern recognition and generation. This suggests that future models will be even more efficient, resulting in more effective *general imitation learning machines*. This is a remarkable phenomenon, considered that it does not require neither gathering more robotics-tailored data, nor actively researching fundamentally novel imitation learning algorithms.

A noteworthy aspect is the ability to effectively process longer sequences as inputs: GPT-2 was limited at 1024 word tokens, while later models saw a 10x-30x increase in a few years. Scaling the context input size of text-pretrained Transformers can help make in-context learning of trajectories more effective, tackling the bottleneck we observed in Fig. 5.

## VI. CONCLUSION

We introduced **Keypoint Action Tokens**, a framework that enables in-context imitation learning of human demonstrations by repurposing large Transformers pretrained on text as general sequence-to-sequence learners. By tokenising both visual inputs and action outputs into a format suitable for text-based Transformers, we demonstrate that we can achieve state-of-the-art results in few-shot imitation learning on a series of challenging everyday tasks. We also analysed what design decisions result in the optimal performance of our method. Our work demonstrates that the progressive evolution of large Transformers pretrained on language, where data is abundant, can unexpectedly benefit the field of robot learning, where data is scarce.

## VII. LIMITATIONS

While achieving superior results in few-shot imitation learning with  $\leq 10 - 20$  demos, KAT does not currently scale as well as Diffusion Policies. We hypothesise that this is due to KAT relying entirely on in-context imitation learning, that can start acting suboptimally when the input sequences (the tokenised inputs and outputs of the demos, and the tokenised new observation) become excessively long. We believe that once  $> 50$  demos are available, it may be beneficial to finetune the pretrained Transformers instead of relying entirely on feed-forward in-context learning.

To tokenise the visual observations, we rely on extracting a predetermined amount of keypoints  $K$ . While we showed that this approach is effective, and even improves the performance of end-to-end vision based Diffusion Policy, the method cannot adapt the number of extracted keypoints to different situations, where e.g. more objects are present. Future work into adaptable and dynamic keypoint extraction may improve the performance of keypoint-based methods, like KAT.

Given a prompt of length  $\mathcal{L}$ , the computational complexity of generating output tokens, based on the classic Transformer architecture [30], is  $\mathcal{O}(\mathcal{L}^2)$ . Therefore, current Transformers will scale poorly to larger datasets of demonstrations in scenarios where online control is fundamental. However, there are several research avenues currently researching an efficient

alternative to classic Transformers [5] or proposing alternatives which are equally effective but linear in complexity [20].

### A. Collecting Demos and Train-Test Distributions

When collecting demos, we randomise the pose of the objects on the table. In particular, we move the objects on a  $50\text{cm} \times 30\text{cm}$  area, and randomise their orientation around the imaginary vertical axis in a 45 degrees range from a "standard orientation". To test for extrapolation at test time, we move objects in a larger area of  $70\text{cm} \times 40\text{cm}$  and rotate them in a 60 degrees range.

For tasks that can be tested for objects generalisation (*Wiping a Plate, Sweep, Scooping, Pick and Pour, Bottle Upright, Pick and Place*) we provide demonstrations using 2 different sets of objects, and then test on both these sets and a third unseen set.

When testing for robustness to distractors, we change our background, which is normally a black cloth, to expose the wooden table underneath, and position an additional set of 2-3 random objects of different classes than the ones involved in the task, randomly positioned as the other objects.

### B. Prompting the Language Model

We mainly use GPT-4 Turbo for our experiments, except when explicitly comparing it to GPT-3 and GPT-2. For GPT-4 and GPT-3 we use the official OpenAI API. For GPT-2, we use the Transformers library by HuggingFace to compute its outputs locally.

While largely receiving tokenised numbers as input, the way GPT-4 is structured requires a "system" message describing what its task is. We therefore briefly describe that GPT-4 is tasked to solve a time series pattern recognition task, receiving examples of sequence-to-sequence data, and needing to find an output sequence given an input sequence that emulates the underlying patterns of the inputs. Notice we *do not mention robotics or tasks anywhere*: the model is only asked to act as a sequence-to-sequence general pattern recognition machine.

In particular, the instruction is

*"You are a pattern generator machine. I will give you a series of patterns with INPUTS and OUTPUTS as examples. Then, you will receive a new INPUTS, and you have to generate OUTPUTS following the pattern that appears in the data. The points are (x,y) coordinates. Only reply with the OUTPUTS numbers."*

When comparing Language Models, we use the same prompt for each, composed of the above, plus the tokenised demonstrations as sequences of keypoints and action tokens, and finally the test time sequence of new keypoint tokens.

### C. Extracting descriptors from DINO, CLIP and R3M

Here we describe in more detail how we extract the dense descriptor maps, given an input observation, to then extract keypoints. For DINO, we use *DINO-ViT*s8, with a stride of 4. We extract the *key* descriptors from the 9th layer. For CLIP, we use the CLIP RN50 version, and extract the output of the fifth-to-last layer of the ResNet50 backbone, a 3 dimensional tensor. Likewise for R3M, we extract the output of the fifth-to-last layer of the ResNet50 backbone, a 3 dimensional tensor.

We explored each of these hyperparameters to maximise the performance of each of these methods.

#### *D. Implementation of Diffusion Policy*

As mentioned in the main paper, we rely on the official implementation by the authors of Diffusion Policy to train it as a baseline.

We adapt the input and output spaces to our tasks, using vision for the end-to-end baseline and the state-based for the KeyAct baseline that takes keypoints as input. We modify mostly the training parameters, such as batch size and epochs, to maximise performance by evaluating both real world results and train/validation losses.