SIMPLE VIDEO CONTEXT SCALING OF ROBOTIC POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Intelligent robotic agents should exhibit strong memory capabilities over very long temporal horizons. Memory can be facilitated by directly feeding video history as context to the policy, but doing so comes with a significant challenge. Video memory spanning more than even just a few seconds quickly becomes computationally difficult: simple token math mandates $O(50k)$ tokens of context just for 1 minute of 1 fps video memory with current standard methods. To address this, in this work we show that by adopting an early-fusion-like model architecture, even very long videos can fit completely and token-efficiently in transformer contexts, changing the token math substantially. On simulated memory benchmark tasks our method, *Scaled Robotic Memory via Early-Fusion* (SRoME), greatly surpasses the token efficiency of history representations compared to baselines, while also being more performant. We also show that train step times and inference times are quick, with very favorable scaling behavior when memory context length scales. In short by tackling the main challenge that plagues video memory policies today, video context length, SRoME makes long video memory policy training easy.

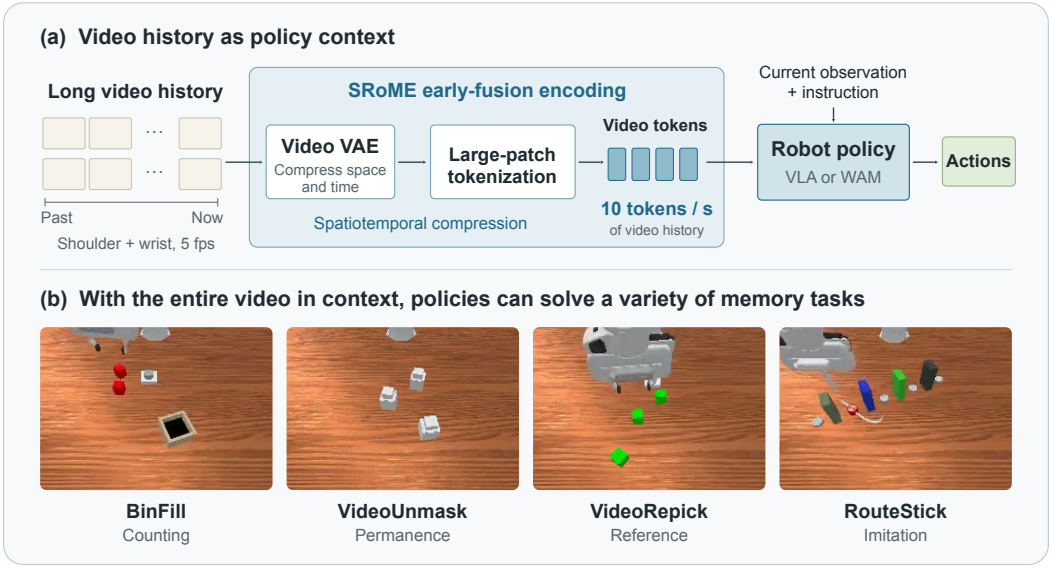

Figure 1: **Scaled Robotic Memory via Early-Fusion (SRoME).** Part (a): Video history is spatiotemporally compressed and tokenized with large patches, yielding an efficient set of video tokens that can be fed to VLA or WAM policies. Part (b): on RoboMME, a simulated memory benchmark, being able to fit the full video history in context enables policies to perform a variety of memory tasks, outperforming baselines while also being more token efficient.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

For robots to complete increasingly intelligent physical tasks in the real-world, they should be able to maintain memory over their past interaction with the world. Imagine a home robot purchased by a user for household manipulation tasks. The user should be able to specify once important contextual details for physical tasks, like where each type of dish should go when unloading a dishwasher, or where tennis racquets are usually kept, and the robot, even weeks or months later, should be able to recall these important contexts from its memory.

Towards enabling this type of long-horizon memory, in this work we study scaling video contextual memory. While memory can also be represented in other modalities, like textual annotations (Lin et al., 2025; Torne et al., 2026), image keyframes (Sridhar et al., 2025; Mark et al., 2026), and maps (Huang et al., 2023; Gu et al., 2023), these representations are all lower-dimensional projections of video into a format that historically prior work found more easily ingestible by policies (due to the difficulties with scaling video context). If we are able to scale video context, robot policies can condition on the most lossless representation of video memory, namely the video itself, and video becomes a complementary conditioning modality to existing modalities like text and keyframes.

Scaling video context for robotic policies has been notoriously difficult due to a token blowup that arises when conditioning on video frames. While robot policies can exist in many forms, the current state-of-the-art robot policy model architectures generally are vision-language-action (VLA) models finetuned from VLMs (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; Li et al., 2024; NVIDIA et al., 2025; Physical Intelligence et al., 2025; Kim et al., 2025; Shukor et al., 2025), or policies bootstrapped off of image or video generation models, often called world-action models (WAMs) (Du et al., 2023; Black et al., 2023; Hu et al., 2024; Shen et al., 2025; Kim et al., 2026; Li et al., 2026; Ye et al., 2026). These models condition on visual information by feeding each individual frame through a semantic visual encoder such as SigLIP2 (Tschannen et al., 2025), or for WAMs through an image or video VAE. Some models do both (Ye et al., 2026). This results in several hundred tokens per image. Concretely, for a 1 fps video with a shoulder and wrist camera, each at resolution 252×448, $\pi_{0.5}$'s (Physical Intelligence et al., 2025) visual encoder produces 1152 video tokens per second. Cosmos3's (NVIDIA, 2026) encoder, a video VAE, produces 56 tokens per second. Scaling models to one minute of 1 fps video memory results in $70k$ tokens for $\pi_{0.5}$ and 3300 tokens for Cosmos3. Scaling to 1 hour results in over $4M$ tokens for $\pi_{0.5}$ and over $200k$ tokens for Cosmos3. These token counts, just for 1 hour of video memory, already are at the max token counts of frontier LLMs, and we desire to scale video memory much further than one hour, e.g., to over days or even longer.

The high token cost means that state of the art robot policies limit the amount of video memory to make training tractable. A great majority of recent robot foundation models do not possess memory at all (Kim et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025; NVIDIA et al., 2025; Kim et al., 2025; Shukor et al., 2025), relying on just one frame for all action predictions. Others that do have memory have usually a minute or less (Brohan et al., 2022; Torne et al., 2026). This suggests that what we desire is a way to reduce the token count needed to fit video history into context.

In this paper we observe that there is a very simple, early fusion style video encoding mechanism that enables long videos to fit entirely and completely into robot policy contexts. Video is highly spatially and temporally redundant. Thus, before tokenizing a video it is sensible to compress the video with a model that can remove these spatio-temporal redundancies. Video VAEs are trained exactly with this objective, and in this work we use a video VAE that is part of a unified understanding and generation visual encoder. Then, by simply using a large patch size, crticially the largest possible patch size that results in a post-patch hidden dimension that is still less than the transformer hidden dimension, it is possible to tokenize the compressed video very efficiently, with no theoretical information loss in this tokenization procedure with respect to the compressed video representation. We show that using this procedure, barring the spatio-temporal redundancies that are discarded by the compressor, entire videos can quite simply be fit into context extremely token efficiently.

While this large patchification mechanism originated in recent work that attempted to scale context length for video generation models (Zhang & Agrawala, 2025), in this work we show that performant robot policies can be constructed using the same early fusion design, and these architectural changes work well with current robot foundation model designs. In particular we train policies from two

robot foundation model families, VLAs and WAMs, with memory encoded by our early fusion spatio-temporally compressed video encoding scheme, which we call *Scaled Robotic Memory via Early-Fusion* (SRoME). On RoboMME (Dai et al., 2026), a simulated robotic memory benchmark, we show that SRoME enables more performant policies in both VLA and WAM model families while also more token-efficiently representing video history. We also show inference and train step time scaling curves, demonstrating that token efficiency is paramount to both training faster, and thus with the same compute being able to train on longer video contexts, as well as inferencing faster.

2 Related Work

Recent work gives robot policies access to past interactions through several forms of memory. MemoryVLA (Shi et al., 2025) retrieves perceptual and semantic features, while MemER (Sridhar et al., 2025) selects earlier keyframes to guide a hierarchical policy. MEM (Torne et al., 2026) combines recent video with longer-term textual memory, using spatial and temporal attention followed by token dropping to reduce the video token count. We study how cheaply policies can condition on the full sampled video history, complementing memory represented in text or keyframes.

Video generation provides a useful starting point for this problem. Video generative models often employ learned autoencoders that compress visual inputs (Rombach et al., 2022; Lu et al., 2025), followed by a patchification procedure that converts these inputs into tokens. Unified multimodal models also show how visual tokens can be processed directly alongside text (Chameleon Team, 2024; Zhou et al., 2024; Deng et al., 2025). Particularly closely related to our work is FramePack (Zhang & Agrawala, 2025), which allocates different representation capacities to past frames to bound the number of tokens it takes to represent long videos. In our work, we study the use of spatio-temporal compression followed by a large-patch-size patchification operator for construction of video contexts for robot policies.

Modern robot policies are often constructed from pretrained models that already process language and visual inputs. For example, RT-2 and OpenVLA finetune VLMs to predict robot actions as tokens (Brohan et al., 2023; Kim et al., 2024), while π_0 and $\pi_{0.5}$ add a continuous action expert to the VLM backbone (Black et al., 2024; Physical Intelligence et al., 2025). Another way to construct a policy is to first predict how the scene should change to complete a task. UniPi and SuSIE follow this recipe, using image or video generative models to predict future observations, and then a separate controller to predict actions that reach the generated goal (Du et al., 2023; Black et al., 2023). Other WAMs, such as Cosmos Policy and LingBot-VA, jointly predict future observations and actions (Kim et al., 2026; Li et al., 2026). In all of these cases, a policy needs access to past observations whenever the current image does not provide enough context to determine what to do. In our work we study how to supply this context as a video, and show that the same video encoding scheme works well for both VLA and WAM architectures. For the VLA, the video history directly informs action prediction. For our WAM, it informs the future image prediction, which in turn tells a goal-conditioned policy what state to reach.

3 Method

In this section we discuss how SRoME facilitates efficient video encoding. Then we describe how modern robot policy architectures, built off of VLA or WAM backbones, can be adapted to consume SRoME's video representation, essentially by simply feeding in all video tokens into the main transformer context.

3.1 Encoding Video History

In most VLM architectures, each image input is fed through a semantic image encoder, such as SigLIP2, yielding a set of visual tokens that are then projected into the language model backbone's context. Instead of this recipe, our method adopts a more early-fusion-like recipe for conditioning on visual information. In early-fusion vision language models, the image or video input is tokenized directly from its raw representation, and the useful visual representations are learned end to end with

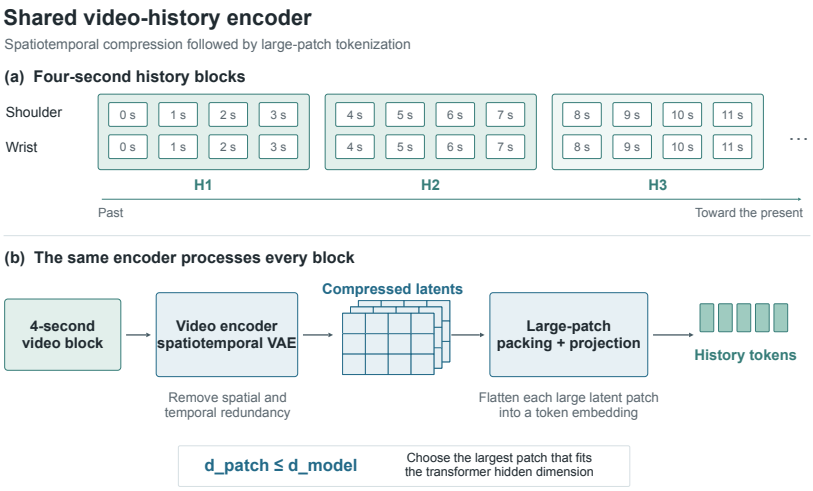

Figure 2: Video is compressed in short clips, then tokenized with large patches. This example shows two camera views sampled at 1 fps.

the model's training objectve (i.e., next token prediction), as opposed to the visual encoder having been pre-trained with a different objective.

We observe that while we can directly tokenize raw video, videos are very highly spatially and temporally correlated. A standard video codec like MPEG might for example reduce a video's file size by an order of magnitude. Especially given that video tokenization is notoriously expensive in terms of token count, we explore first spatio-temporally compression the video, before performing early fusion tokenization.

We thus look for a mechanism to, like MPEG, reduce the *file size*, or effectively the bits in the video representation, of a video input, critically in doing so producing features that are amenable to being consumed by a neural network. Fortunately the video generative modeling community has a recipe for exactly this: training a video VAE that spatio-temporally compresses video into a lower-dimensional latent space, effectively performing compression. While we could use any such video VAE, in this work we use the AToken unified visual encoder (Lu et al., 2025), as in addition to spatio-temporal compression the encoder is trained with understanding losses. In practice though, any reasonable video VAE can be used. These VAEs reduce spatio-temporal redundancies within a video-clip. Concretely with Atoken, each video "clip" is a 4-frame sequence of images. This mechanism is very similar to the P and I frames in classical video codecs like MPEG only compressing locally, rather than globally, as even with local compression a substantial amount of spatio-temporal redundancy can be removed.

Let the compressive encoder output a latent tensor with shape $T_z \times H_z \times W_z \times c$. We partition its spatial grid into $p \times p$ patches, and concatenate the entries in each patch. The history token count and latent embedding dimension then become

$$N_H = T_z \frac{H_z W_z}{p^2}, \qquad d_{\text{patch}} = cp^2. \tag{1}$$

Normal use of video VAEs, for example in video generation, deliberately use a small patch size, often 1×1 or 2×2, following (Peebles & Xie, 2023). The reason for doing this is to intentionally increase the number of tokens representing the video so as to increase computation, and therefore downstream generation performance. We posit that when video is being used as context, for example as memory context for a robotic policy, it is perhaps not necessary to intentionally increase the flop count in this manner, as the conditioning information is not being generated. We thus take an opposite approach to (Peebles & Xie, 2023), making the patch size as large as we can make it to have the most efficient tokenization scheme. We choose large patches whose flattened latent dimension fits within the transformer's hidden dimension. Doubling p reduces the token count by a factor of four, while increasing each token's hidden dimension by four. This trick makes use of the hidden

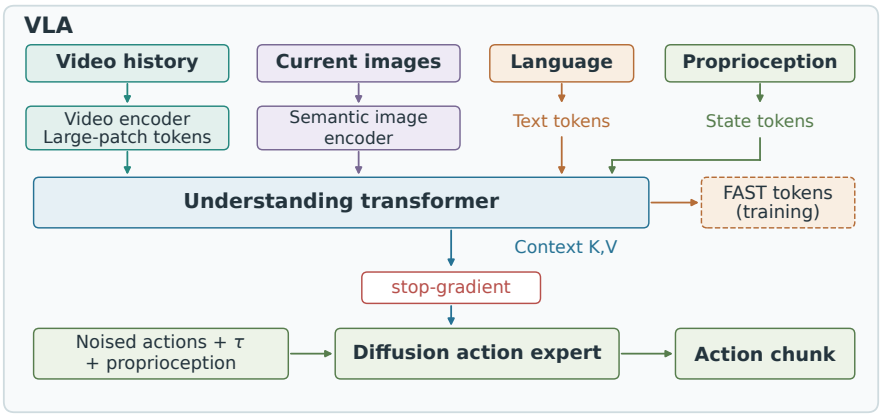

Figure 3: SRoME VLA. Compressed video tokens are fed directly into the VLA context, alongside the current observation and task instruction.

dimension capacity already available in the transformer backbone to shorten the sequence length, and this "packing" of tokens is lossless, in that it discards no information from the compressed video representation.

3.2 Attention Masks for Video Context

Recall that the learned video VAE spatio-temporally compresses video *clips*. To accomodate this, and to still enable efficient inference with video history via key-value caching, we employ a blockwise-causal attention mask for history tokens in all models that consume SRoME's video memory. In particular within a video clip block, tokens can fully attend to one another, and across blocks they attend only causally.

This causal structure is particularly useful in inference, as robot policies, more so than LLMs or VLMs, must obey inference latency constraints especially for real-world control. Once a video clip context has been encoded, its KVs can be cached enabling fast inference with increasingly long video contexts.

3.3 Adapting VLAs with SRoME's Video Encoder

In this section we discuss how modern VLA architectures, which typically just consume one image of visual context, can be adapted to condition on very long video contexts via SRoME.

The adaptation is quite simple. We keep the VLA architecture, which typically does not employ an early fusion design, as is, and simply insert in early-fusion style the video context tokens to the VLA's LLM backbone context. As mentioned in the previous section these tokens are attended to blockwise-caussally, and the final transformer sequence consists of language task description tokens, video tokens, image tokens of the current image that is fed through the VLA's usual image encoder, and action tokens.

Following prior work in this paper we employ VLAs with knowledge insulation (Pertsch et al., 2025; Driess et al., 2025) (KI). KI trains policies with next token prediction via actions discretized via the FAST action tokenizer (Pertsch et al., 2025), and trains a flow-matching action expert that can condition on intermediary VLA features but one that does not backpropogate gradients through the rest of the model. So for a conditioning context C_t and FAST-encoded target chunk D^*, the objective is

$$\mathcal{L}_{\text{FAST}} = -\sum_j \log p_\theta(D^*_j \mid C_t, D^*_{<j}). \tag{2}$$

The action expert learns continuous actions through flow matching. Given actions a, Gaussian noise ϵ, and time τ, its input is $a^\tau = (1-\tau)a + \tau\epsilon$, and its loss is

$$\mathcal{L}_{\text{act}} = \mathbb{E}\left[\left\| v_\psi\left(a^\tau, \tau; \text{sg}(\text{KV}_\theta(C_t))\right) - (\epsilon - a)\right\|^2\right]. \tag{3}$$

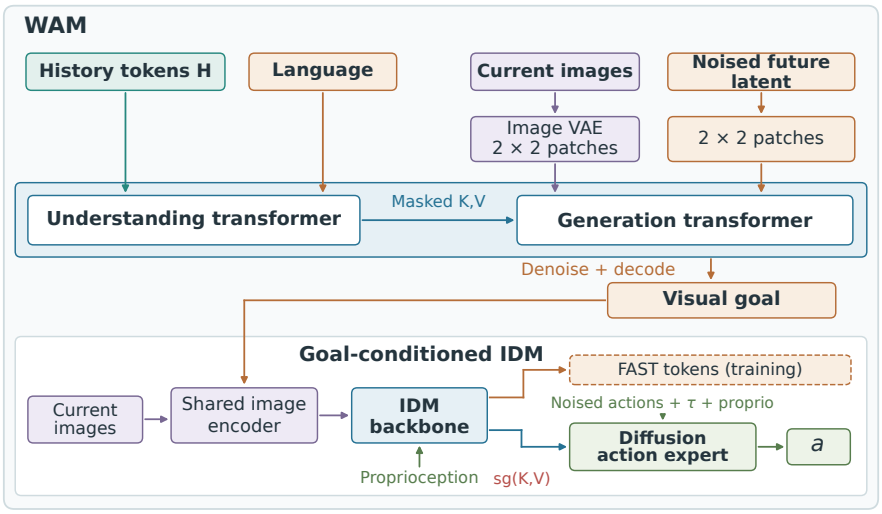

Figure 4: SRoME WAM. The model predicts a future image from video history, and a goal-conditioned policy predicts actions to reach it.

Here sg is the stop-gradient operator that stops the action flow loss from updating the backbone, which is instead supervised via next token prediction with FAST. At inference, we use the action expert only, discarding the FAST tokens.

3.4 Adapting WAMs with SRoME's Video Encoder

Similar to VLAs, we show in this section how world-action models (WAMs), which either generate a future image of how the world should evolve and then predict actions conditioned on the current world state and this future prediction, or simultanesouly predict future images/video and actions, can be adapted to consume long video history with SRoME.

Given the high diversity of WAM architectures, we pick one representative example, namely the image generating model from $\pi_{0.7}$. Concretely, we construct a WAM from the BAGEL mixture-of-transformers model (figure 4). BAGEL is a model with two 7B parameter components: an understanding expert processes text and potentially visual information, and a generation expert, which is trained with flow matching, produces images conditioned on language and/or the current image (i.e., in an image-editing setup). Languge tokens and compressed video tokens are processed by the understanding half of the BAGEL model, and the current image and noised future image are processed by the generation half. Critically in the generation half the current image and noised future image are fed through BAGEL's own VAE, and then patchified with a fine 2x2 patchification operator. This introduces an assymetry of sorts, where SRoME tokens are processed with a large patch size, and generation tokens a small patch size. This is effectively a best-of-both-worlds arrangement, where video context can scale via its efficient tokenization, and generation performance can remain strong due to the increased token count of generation tokens.

The two transformers interact through attention. Language attends causally only to language tokens, and history follows the block-causal attention pattern mentioned earlier. Current-image tokens attend to only other current-image tokens, and future-image tokens can attend to everything. The future image prediction can thus depend on the instruction, video history, and current observation.

The model is trained with flow matching. At inference, it denoises a latent from Gaussian noise generating a future image prediction G_t. This generated image is then fed to a separate goal-conditioned policy/inverse dynamics model (IDM), which predicts actions from (I_t, G_t, q_t), i.e., the current image, generated goal, and robot proprioception information. We construct the IDM to exhibit a similar architecture as the VLA architecture in the previous section, using FAST and KI, with the main difference being that rather than SRoME history, the IDM takes as input the generated goal image.

Table 1: Task success on RoboMME (%), evaluated over 50 episodes per task.

Suite	Task	SRoME WAM		SRoME VLA	BAGEL
		Patch 8	Patch 4	Direct action	MEM
Counting	BinFill	**48**	**48**	6	2
	PickXtimes	74	**80**	18	6
	SwingXtimes	**28**	14	4	0
	StopCube	6	0	**10**	6
Permanence	ButtonUnmask	**6**	**6**	4	4
	ButtonUnmaskSwap	0	0	**2**	0
	VideoUnmask	**22**	**22**	6	16
	VideoUnmaskSwap	**16**	14	8	14
Reference	PickHighlight	32	**34**	2	4
	VideoRepick	**28**	20	2	4
	MoveCube	50	**52**	36	24
	InsertPeg	**4**	2	0	0
Imitation	PatternLock	**14**	**14**	12	10
	RouteStick	8	**12**	**12**	**12**
	VideoPlaceButton	**28**	26	10	16
	VideoPlaceOrder	**24**	20	12	16
Overall success (%)		**24.25**	22.75	9.00	8.38
Successful episodes / 800		194	182	72	67

4 Experimental Results on Memory Tasks

4.1 Setup

We evaluate on RoboMME, which is a simulated memory benchmark containing 16 simulated manipulation tasks spanning counting, object permanence, reference, and imitation (Dai et al., 2026). For each task, the training set contains of 100 demonstrations, resuting in 1,600 demonstrations total in the benchmark.

We train VLA and WAM models with history directly from their corresponding model initializations. The WAM is initialized from the full 14B BAGEL model (Deng et al., 2025), and the VLA as well as the IDM are initialized from the "VLM half" of BAGEL, i.e., just the understanding transformer. The WAM therefore has additional generative capacity, but all policies are trained on the same robot training data, namely the 1600 demonstrations.

Our main baseline is MEM (Torne et al., 2026), which we construct using the 7B understanding half of BAGEL by modifying its semantic visual encoder to process video with alternating spatial and temporal attention, followed by token dropping, as in MEM. We also compare a number of ablations, namely whether the VLA or WAM architecture is more performant, and also with the WAM, whether intentionally decreasing the patch size beyond its maximum value, which intentionally increases the number of video tokens, is effective or helps in any way.

The simulator runs at 20 Hz, and we sample history at 5 Hz. Visual observations consist of a shoulder camera view and a wrist camera view, which we stack into one video frame. Each stacked RGB clip has shape $4 \times 512 \times 256 \times 3$, producing a $32 \times 16 \times 48$ latent grid after processing by Atoken. Using a patch size of eight results in eight tokens per clip with a hidden dimension of 3,072, under BAGEL's hidden dimension. Using patch size four results in 32 tokens with a latent dim of 768.

Models are trained to predict the next 20 timestep of action commands, and for the WAM model, the visual target is the observation 20 control steps ahead. We train models with batch size 128, using Adam with learning rate 10^{-4}, 1,000 warmup updates, and a total of 60,000 gradient steps. Further implementation details can be found in the Appendix A.

4.2 Task Success

Table 1 reports the test-set policy performance evaluation over 50 episodes per task.

Table 2: History tokens for two-camera video at 5 fps. MEM counts are before token dropping; SRoME counts are after compression and patchification.

Representation	4 frames/view	1 min	10 min	1 h
BAGEL-MEM (before dropping)	2,048	153,600	1,536,000	9,216,000
SRoME, 4×4 patches	32	2,400	24,000	144,000
SRoME, 8×8 patches	**8**	**600**	**6,000**	**36,000**

The patch-eight WAM obtains 24.25% overall success, compared to 9.00% for direct action and 8.38% for BAGEL-MEM. This is an improvement of 15.25 and 15.88 percentage points, respectively. The direct-action model is slightly better than BAGEL-MEM on average despite inserting compressed history directly into the language context.

The gains are particularly large on several tasks where the present image cannot reveal the required action by itself. On BinFill, both WAM variants obtain 48% success, compared to 6% for direct action and 2% for BAGEL-MEM. On PickXtimes, the WAM variants reach 74% and 80%, versus 18% and 6%. On PickHighlight, they obtain 32% and 34%, versus 2% and 4%. These tasks require remembering prior events or transient visual cues. The results show that the packed representation can retain useful information for such behaviors.

4.3 Does a Smaller Patch Help?

The patch-four WAM uses four times as many history tokens as patch eight, intentionally increasing compute, but obtains 22.75% overall success rather than 24.25%. This suggests that unlike in image or video generation, video as context need not necesarily be expressed in higher-compute (i.e., more token) representations. Video tokenization can remain efficient and at the same time performant.

4.4 Video Token Efficiency

A short language-model context does not necessarily mean that conditioning on video history is cheap. In MEM, the history is processed within the visual encoder, and token dropping keeps only the current-frame features for the language model. However, in our BAGEL-MEM baseline, each camera frame is first tokenized into a 16×16 grid, and all of these historical tokens are processed through the visual encoder's 26 layers before they are dropped. With two cameras sampled at 5 fps, this amounts to 2,560 vision tokens for every second of history. So even though MEM keeps the language-model context short, its visual encoder still has to process a very large number of tokens as the history grows.

Table 2 compares this cost with SRoME for the same video histories. Concretely, ten minutes of video require 1,536,000 history tokens in MEM's visual encoder, while SRoME with patch size eight uses just 6,000 tokens, a $256\times$ reduction in token count. Even the smaller patch size of four uses $64\times$ fewer tokens than MEM. At one hour, the difference grows to over nine million tokens for MEM versus 36,000 for SRoME with patch size eight. The reason for this difference is that SRoME compresses short video clips before constructing the policy's video context. Thus, the policy can condition on the full sampled history through a much smaller set of tokens. As the preceding ablation shows, this efficient representation also remains performant.

These counts measure the history representation, excluding current observations and padding; the longer durations show how token counts scale beyond the benchmark episodes. The token-count reduction does not directly specify a runtime speedup, since MEM processes history in its visual encoder while SRoME feeds compressed history tokens to the policy transformer. In the next section we measure how these architectural differences affect training and inference times.

5 Training and Inference Scaling

We next measure how the token savings translate into training and inference times as the amount of video in context grows (Figure 5). For these measurements, we sample video at 1 fps.

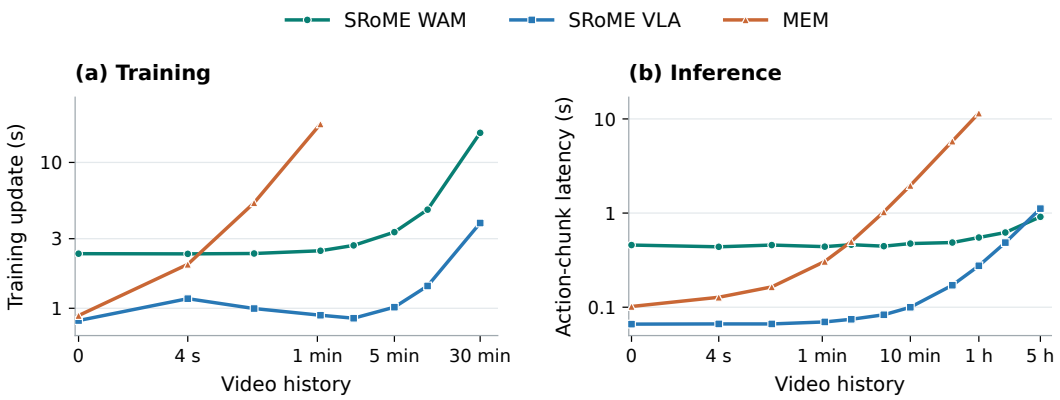

Figure 5: Training update times on 128 TPU v5e chips with batch size 128 (left). Inference times on up to four H100 80GB GPUs with batch size one and cached history (right). Details are in Appendix A.1.

With 64 seconds of history, a training update takes 0.90 seconds for SRoME VLA and 2.48 seconds for SRoME WAM, compared with 18.35 seconds for MEM. Both SRoME variants can be trained with 30 minutes of history, whereas MEM runs out of memory at 128 seconds in this setup. The reduction in tokens thus allows us to train faster, and also to train with much longer videos within the same hardware budget. These times include the forward pass, backward pass, and parameter updates. For the WAM, we time training the model that predicts future images. We train the IDM separately, so its training time is not included in the plot.

During inference, the policy needs to use its video history each time it predicts actions, so re-encoding the entire video at every control step would quickly become expensive. Our attention mask lets us avoid this. Once a clip has been encoded, its representation does not change when new observations arrive, because history tokens do not attend to later clips. We can therefore cache each clip's keys and values and reuse them for subsequent predictions. The policy still attends to the history when predicting actions, but does not have to encode the full video again. Concretely, with 30 minutes in context, SRoME VLA predicts an action chunk in 0.17 seconds and SRoME WAM in 0.49 seconds, compared with 5.81 seconds for MEM. Even at five hours, the SRoME latencies are 1.11 and 0.91 seconds, respectively. Here we time action prediction with the history already cached. Building the initial cache and adding new clips to it are excluded from these measurements. Appendix A.1 gives the full timing setup.

6 Discussion and Conclusion

The WAM performs better than the VLA even though both receive the same video history. We posit that predicting future images helps make better use of the available robot data, following the motivation of prior work on generative policies (Du et al., 2023; Black et al., 2023; Li et al., 2026). In particular, each demonstration provides images of how the world changes, in addition to the actions themselves. The WAM can use these images as supervision, and use its generative pretraining to predict a goal that the IDM then learns to reach.

In this work we show that a simple change to video tokenization makes long video contexts more practical for robot policies. By first compressing video in space and time, and then using large patches, SRoME represents the full sampled history with relatively few tokens. On RoboMME, increasing the history token count fourfold does not improve aggregate success: our most efficient patch size is also the most performant. This representation works well with both VLA and WAM architectures, allowing policies to use video memory while substantially reducing the cost of conditioning on it.

AI USE STATEMENT

We used generative AI to assist with research code and experiments, as well as light paper editing.

ETHICS STATEMENT

We evaluate on an existing simulation benchmark, and we do not anticipate any ethical concerns with this.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 describe the method and experiments, and appendices A and B give implementation details, timing protocols, and attention masks. Together this should help towards reproducibility.

REFERENCES

Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. *arXiv preprint arXiv:2310.10639*, 2023. URL https://arxiv.org/abs/2310.10639.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. URL https://arxiv.org/abs/2212.06817.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. URL https://arxiv.org/abs/2307.15818.

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024. URL https://arxiv.org/abs/2405.09818.

Yinpei Dai, Hongze Fu, Jayjun Lee, Yuejiang Liu, Haoran Zhang, Jianing Yang, Chelsea Finn, Nima Fazeli, and Joyce Chai. RoboMME: Benchmarking and understanding memory for robotic generalist policies. *arXiv preprint arXiv:2603.04639*, 2026. URL https://arxiv.org/abs/2603.04639.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, Guang Shi, and Haoqi Fan. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025. URL https://arxiv.org/abs/2505.14683.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint arXiv:2505.23705*, 2025. URL https://arxiv.org/abs/2505.23705.

Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *arXiv preprint arXiv:2302.00111*, 2023. URL https://arxiv.org/abs/2302.00111.

Qiao Gu, Alihusein Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, Chuang Gan, Celso Miguel de Melo, Joshua B. Tenenbaum, Antonio Torralba, Florian Shkurti, and Liam Paull. ConceptGraphs: Open-vocabulary 3D scene graphs for perception and planning. *arXiv preprint arXiv:2309.16650*, 2023. URL https://arxiv.org/abs/2309.16650.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. *arXiv preprint arXiv:2412.14803*, 2024. URL https://arxiv.org/abs/2412.14803.

Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023. URL https://arxiv.org/abs/2210.05714.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025. URL https://arxiv.org/abs/2502.19645.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026. URL https://arxiv.org/abs/2601.16163.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026. URL https://arxiv.org/abs/2601.21998.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. CogACT: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024. URL https://arxiv.org/abs/2411.19650.

Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. OneTwoVLA: A unified vision-language-action model with adaptive reasoning. *arXiv preprint arXiv:2505.11917*, 2025. URL https://arxiv.org/abs/2505.11917.

Jiasen Lu, Liangchen Song, Mingze Xu, Byeongjoo Ahn, Yanjun Wang, Chen Chen, Afshin Dehghan, and Yinfei Yang. AToken: A unified tokenizer for vision. *arXiv preprint arXiv:2509.14476*, 2025. URL https://arxiv.org/abs/2509.14476.

Max Sobol Mark, Jacky Liang, Maria Attarian, Chuyuan Fu, Debidatta Dwibedi, Dhruv Shah, and Aviral Kumar. BPP: Long-context robot imitation learning by focusing on key history frames. *arXiv preprint arXiv:2602.15010*, 2026. URL https://arxiv.org/abs/2602.15010.

NVIDIA. Cosmos 3: Omnimodal world models for physical AI. *arXiv preprint arXiv:2606.02800*, 2026. URL https://arxiv.org/abs/2606.02800.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. URL https://arxiv.org/abs/2503.14734.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025. URL https://arxiv.org/abs/2501.09747.

Physical Intelligence et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. URL https://arxiv.org/abs/2112.10752.

Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. VideoVLA: Video generators can be generalizable robot manipulators. *arXiv preprint arXiv:2512.06963*, 2025. URL https://arxiv.org/abs/2512.06963.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025. URL https://arxiv.org/abs/2508.19236.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025. URL https://arxiv.org/abs/2506.01844.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. MemER: Scaling up memory for robot control via experience retrieval. *arXiv preprint arXiv:2510.20328*, 2025. URL https://arxiv.org/abs/2510.20328.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. MEM: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv:2603.03596*, 2026. URL https://arxiv.org/abs/2603.03596.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025. URL https://arxiv.org/abs/2502.14786.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi Jim Fan, and Joel Jang. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. URL https://arxiv.org/abs/2602.15922.

Lvmin Zhang and Maneesh Agrawala. Frame context packing and drift prevention in next-frame-prediction video diffusion models. *arXiv preprint arXiv:2504.12626*, 2025. URL https://arxiv.org/abs/2504.12626.

Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. *arXiv preprint arXiv:2408.11039*, 2024. URL https://arxiv.org/abs/2408.11039.

A Implementation Details

Here we give additional details of the action models and their training. The VLA and the WAM's goal-conditioned policy use the same action prediction interface, so the main difference between them is the information from which they must determine the next action.

A shared action interface. The direct-action model and the goal-conditioned IDM both use BAGEL's understanding architecture with a knowledge-insulated action expert. For the direct-action model, the conditioning consists of the task instruction, compressed video history, current images, and proprioception. The IDM instead receives current and goal images, proprioception, and a fixed goal-reaching prompt. Thus the VLA must use the history itself to decide how to act, while the IDM receives the visual goal produced by the WAM's image predictor. Although the two action models have the same architecture, we train their weights in separate runs. The continuous action expert has 28 layers and width 1,024, with key and value head dimensions matching those of the understanding backbone.

Image representations. Current and goal images are resized from the simulator's native 256×256 resolution to 224×224 before being passed to the semantic vision encoder. With 14×14 image patches, this produces 256 image tokens per camera. History follows the separate compression path described in Section 3.1: we retain the native camera resolution, stack the two camera views, and pass the resulting video through the compressive encoder. This allows the model to retain the usual semantic image representation for its current observation, while using the much shorter latent-patch representation for its history.

Action representation and supervision. Each training example predicts a chunk of 20 native control commands following the current observation. Actions consist of joint targets relative to the current configuration and a gripper command. The model uses eight action coordinates, including the gripper representation, and masks coordinates according to physical validity. We compute normalization statistics using only the training data, and use the same normalized action chunk for FAST supervision and continuous flow matching. Near the end of a demonstration, fewer than 20 recorded commands may remain. We fill the rest of the chunk by repeating the final recorded command, so these examples also supervise the demonstrated terminal behavior. Future-image targets are similarly clipped to the recorded episode endpoint.

The FAST interface appends 2,048 action codes and beginning/end markers to BAGEL's 152,064 pretrained vocabulary entries. We train the understanding weights with a separate causal FAST pass. In particular, the teacher-forced FAST hidden states never enter the conditioning cache used by the continuous action expert. This separation matters because FAST provides supervision during training, whereas the continuous expert must predict the action chunk at inference without access to the target action tokens.

Optimization. We use Adam with $\beta_1 = 0.9$, $\beta_2 = 0.95$, gradient clipping at norm one, and no weight decay. Model parameters use BF16, while optimizer moments and exponential moving averages use FP32. For action flow matching, we sample the interpolation time as $\tau = 0.001 + 0.999u$, where $u \sim \mathrm{Beta}(1.5, 1)$. The learning rate, batch size, warmup, and total number of training updates are given in Section 4.

A.1 Timing Measurements

The timing study measures how the cost of computation changes as more video is placed in context. We use synthetic inputs with the full tensor shapes of each model, so the measurements exercise the model computation without depending on data loading. All timings are taken after warmup, with device synchronization, and exclude compilation. The reported values are medians for the fastest measured layout that passed the numerical checks at each history duration.

For training, we use two 64-chip TPU v5e slices, for a total of 128 chips, with global batch size 128. Inputs are already resident on the devices. Each timed update includes the forward and backward passes, the optimizer update, and the exponential moving average update. The WAM measurement

includes the frozen image-VAE computation associated with training the image predictor, but excludes training the IDM, which is performed in a separate run. Thus its training curve describes the cost of learning to predict images from history.

For inference, we use batch size one and predict chunks of 60 actions. For positive history durations, each measurement starts with a nonzero synthetic history cache at the model's native tensor shapes. We exclude the initial history prefill, appending new history to the cache, and robot communication overhead. This isolates the cost of producing the next prediction once the history is available. We evaluate layouts using up to four H100 GPUs, selecting the fastest numerically qualified measurement at each duration.

Action denoising uses 20 Euler steps. Image prediction uses eight midpoint steps, each requiring two network evaluations, for a total of 16 evaluations. The reported WAM inference latency is the sum of the median image-prediction latency and the median IDM latency. It therefore accounts for both stages of producing an action chunk, while remaining a sum of separately measured component medians rather than a directly measured end-to-end median.

In Figure 5, each curve ends at the longest context for which a measurement completed successfully. Positive history durations and both time axes use logarithmic spacing, with zero history plotted separately. The curves measure computational scaling; they do not establish task performance at the longer durations.

B ATTENTION MASKS

Figure 6 shows the attention masks for the main token groups in our BAGEL simulation implementations. A row specifies which keys and values its query tokens can read. We omit image delimiters and demonstration-boundary markers to make the dependencies easier to see, and all entries are additionally gated by valid-token masks. The groups are arranged for readability; their physical order in the input sequence can differ without changing which attention connections are allowed.

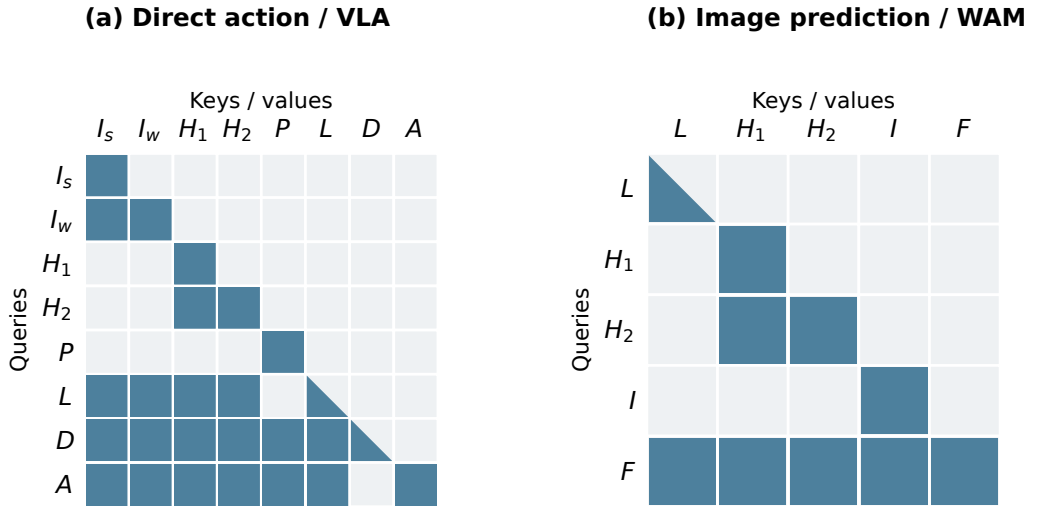

Figure 6: Attention masks for the VLA (left) and WAM image predictor (right). Each row shows which token groups its queries can attend to.

In the VLA mask, I_s and I_w denote current shoulder and wrist image groups; H_1, H_2 denote history clips; P is the numeric proprioception token; L is the language/state text prefix; and D and A are FAST and continuous action tokens. In the WAM mask, L denotes language, H_i denotes a history clip, I denotes current-image tokens, and F denotes noised future-image tokens. Diagonal triangles indicate causal attention within a group, while full diagonal blocks indicate bidirectional attention. We show both action branches together to describe their conditioning, although FAST supervision uses a separate training pass.

History attention. Both models use the same blockwise-causal structure for video history. Tokens within a four-frame compression block can attend to one another bidirectionally, since they represent a clip that has already been jointly encoded. Across clips, each history block can read only itself and earlier history blocks. History does not attend to language, current images, or prediction targets. As a result, adding another clip does not change the information available to previously encoded history tokens, allowing their keys and values to be reused from a cache. The model can still use information from the full observed history when making a prediction; the mask restricts how that history is encoded, rather than truncating which past clips are available to the prediction.

Direct-action model and IDM. Current-image attention is bidirectional within each camera group and causal across camera groups. The separate numeric proprioception token attends only to itself. Language tokens can read the current images, history, and their causal text prefix, which also contains a discretized representation of state. The model therefore receives state both through this text representation and through the separate numeric token.

The FAST and continuous action branches can each read all of this conditioning. Within the FAST branch, attention is autoregressive so that each target token is predicted from earlier action tokens. Within the continuous branch, action tokens attend bidirectionally, allowing the expert to denoise the action chunk jointly. These attention connections specify what information each branch can use. Knowledge insulation imposes an additional constraint on training: we detach only the context keys

and values passed to the continuous action expert, so its flow-matching loss does not update the understanding backbone. Detachment does not prevent the action expert from reading the context. The backbone instead learns from the FAST loss, and only the continuous action branch is used at inference. For the IDM, current and goal images replace the direct-action model's two-image input, and there is no history prefix.

World-action model. In the image predictor, the understanding transformer processes language and history, while the generation transformer processes current and noised future images. Language attends causally within text and does not read the other token groups. History uses the blockwise mask described above. Current-image tokens attend only within the paired-camera current-image group. Future-image tokens can attend to all valid conditioning tokens and bidirectionally to one another. Thus each future token can use the instruction, the entire observed history, and the current observation to predict how the scene should change, while none of these conditioning representations can read the future target.