Visual Encoders for Data-Efficient Imitation Learning in Modern Video Games

Lukas Schäfer*
Microsoft Research
Cambridge, United Kingdom

Logan Jones
Microsoft Gaming
Redmond, United States

Anssi Kanervisto
Microsoft Research
Cambridge, United Kingdom

Yuhan Cao
Microsoft Research
Cambridge, United Kingdom

Tabish Rashid
Microsoft Research
Cambridge, United Kingdom

Raluca Georgescu
Microsoft Research
Cambridge, United Kingdom

David Bignell
Microsoft Research
Cambridge, United Kingdom

Siddhartha Sen
Microsoft Research
New York, United States

Andrea Treviño Gavito
Microsoft Gaming
Redmond, United States

Sam Devlin
Microsoft Research
Cambridge, United Kingdom

ABSTRACT

Video games have served as useful benchmarks for the decision-making community, but going beyond Atari games towards modern games has been prohibitively expensive for the vast majority of the research community. Prior work in modern video games typically relied on game-specific integration to obtain game features and enable online training, or on existing large datasets. An alternative approach is to train agents using imitation learning to play video games purely from images. However, this setting poses a fundamental question: which visual encoders obtain representations that retain information critical for decision making? To answer this question, we conduct a systematic study of imitation learning with publicly available pre-trained visual encoders compared to the typical task-specific end-to-end training approach in Minecraft, Counter-Strike: Global Offensive, and Minecraft Dungeons. Our results show that end-to-end training can be effective with comparably low-resolution images and only minutes of demonstrations, but significant improvements can be gained by utilising pre-trained encoders such as DINOv2 depending on the game. In addition to enabling effective decision making, we show that pre-trained encoders can make decision-making research in video games more accessible by significantly reducing the cost of training.

KEYWORDS

Imitation Learning, Visual Encoders, Video Games

1 INTRODUCTION

Video games have served as useful benchmarks for the decision-making community, training agents in complex games using reinforcement learning (RL) [2, 37, 39], imitation learning (IL) [16, 26, 32], or a combination of both paradigms [1, 9]. Beyond representing a valuable benchmark for complex decision-making tasks, video games represent a vast entertainment industry with many commercial applications of AI agents, including in game development, game testing or game design [10, 14].

Prior research in video games often necessitated close game-specific integration to obtain features and establish a scalable interface for training agents. However, game-specific integration introduces significant cost and requires domain expertise and engineering efforts. To enable decision-making agents for video games without depending on game-specific integration, we focus on training agents to play video games in a human-like manner, receiving only images from the game and producing actions corresponding to controller joystick and button inputs. This framework allows us to train agents entirely offline with behaviour cloning (BC), utilising previously collected human gameplay demonstrations, and evaluate agents without the need for game-specific integration. However, efficient training of agents from video game images necessitates a lower-dimensional representation of high-dimensional images. This motivates our main research question:

Which visual encoders learn representations that retain information for data-efficient decision making in modern video games?

To answer this question, we conduct a comprehensive empirical study of 22 visual encoders in three modern video games: Minecraft [11], Counter-Strike: Global Offensive (CS:GO) [26], and Minecraft Dungeons. First, we examine which visual encoders enable effective decision-making when trained end-to-end from the BC loss. Considered encoders vary in network architecture (ResNets [12, 20] or ViTs [7, 33]), image size, and the application of image augmentations. Second, we study 10 visual encoders that are pre-trained on large datasets of diverse real-world images. These pre-trained encoders promise generalisable representation without additional training and have shown promise in decision-making tasks [22, 24] but it is currently unclear whether these findings transfer to video games that represent a significant domain shift from the real-world images these encoders have been trained on. We identify four categories of pre-trained visual encoders and train

*Contact at lukas.schaefer@microsoft.com.

Proc. of the Adaptive and Learning Agents Workshop (ALA 2025), Avalos, Aydeniz, Müller, Mohammedalamen (eds.), May 19 – 20, 2025, Detroit, Michigan, USA, ala-workshop.github.io. 2025.

agents using the representations from 10 different pre-trained encoders spanning all these categories: self-supervised trained encoders, language-contrastive trained encoders, supervised trained encoders, and reconstruction trained encoders. Third, motivated by the cost of collecting human gameplay data, we investigate the efficacy of these visual encoders for decision making when trained on smaller datasets. Our findings can be summaries as follows:

- End-to-end trained encoders: small 128×128 images are sufficient to enable decision-making in games and image augmentation can significantly improve the performance.
- Pre-trained encoders: DINOv2 [23] is among the best pre-trained encoder across all games and outperforms end-to-end trained encoders in Minecraft and Minecraft Dungeons.
- Data efficiency: capable decision-making agents can be trained with as little as 5 minutes of human gameplay data.
- Training cost: pre-trained encoders can significantly reduce the cost of training in terms of time and memory requirements through pre-computing embeddings on the dataset.

Our approach and study provides a general framework for training decision-making agents from visual inputs in modern video games, and contributes valuable findings that facilitate further research. In particular, the strong performance of pre-trained encoders alongside their comparably little training cost is encouraging and significantly reduces the barrier of entry to conduct research on decision-making agents in video games.

2 RELATED WORK

Learning Agents in Video Games. Video games have commonly served as benchmarks for decision-making agents, but training these agents typically assumed access to a programmatic interface for online training using reinforcement learning [2, 37] or large quantities of expert demonstrations used for offline imitation and supervised learning [1, 9, 28]. Recently, a plethora of work proposed to leverage pre-trained foundation models such as large language models as decision-making agents. The foundation models are either frozen or fine-tuned to directly act within their environment [4, 19, 34, 38] or to collect training data [3], but these works all rely on extensive datasets, game-specific integration, or both. Similar to our approach, Pearce and Zhu [26] and Kanervisto et al. [16] evaluate imitation learning agents in videos games without programmatic interfaces. However, their studies only considered few and relatively simple visual encoders.

Visual Encoders for Decision-Making in Robotics. Prior research has conducted studies into the efficacy of visual encoders for imitation learning or reinforcement learning in robotic domains [22, 24, 30, 41]. Most of these studies found that pre-trained encoders enable better generalisation and performance in decision-making agents than end-to-end trained visual encoders trained on smaller, task-specific data sets. However, we would like to note that not all findings align with these results [30]. These seemingly contradictory findings indicate that the question of which visual encoder works best depends for decision making is nuanced with the answer strongly depending on the underlying algorithm and setting. We further highlight that the images in the majority of robotics tasks and datasets strongly resemble the real world. This stands in contrast to video games that might feature highly stylised

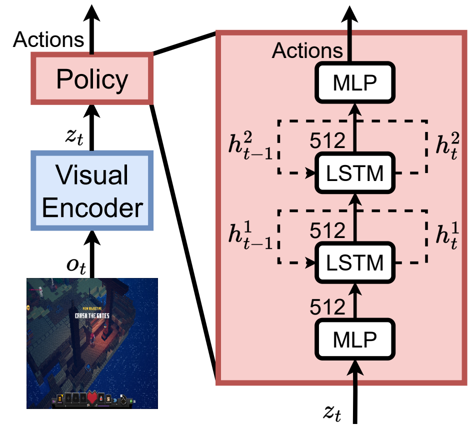


Figure 1: Illustration of the agent architecture. We consider end-to-end trained and pre-trained visual encoders.

images. Therefore, it remains uncertain how these findings from robotics tasks translate to the realm of video games that represents a strong domain shift from the training data of pre-trained visual encoders. Our study seeks to bridge this gap.

Visual Encoders for Video Games. In the context of video games, pre-trained visual models have been employed to extract visual representations that differentiate between genres and styles [36], indicating their ability to detect relevant features in games. However, domain-specific models trained using self-supervised representation learning techniques can yield higher-quality representations than certain pre-trained visual encoders [35]. Our study expands upon previous experiments[1] by concentrating on modern video games and examining a broad spectrum of recent pre-trained and end-to-end trained visual encoder architectures.

3 IMITATION LEARNING FOR VIDEO GAMES FROM PIXELS

Behaviour cloning (BC) is an imitation learning approach that trains agents through supervised learning on a dataset of demonstrations, denoted as $\mathcal{D} = \{(o_1, a_1), \ldots, (o_N, a_N)\}$, where N represents the total number of samples in the dataset. Each demonstration comprises a sequence of tuples (o, a) of an image observation o and a chosen action a. Using this data, a policy $\pi(a \mid o; \theta)$ is trained to mimic the distribution of actions within $\mathcal{D}$ by minimising the loss

$$\mathcal{L}(\theta) = \mathbb{E}_{(o,a)\sim\mathcal{D},\hat{a}\sim\pi(\cdot\mid o;\theta)} \left[l(a, \hat{a}) \right]$$

where l measures the discrepancy between the "true" action a and the policy's sampled action $\hat{a}$. For continuous and discrete actions, we use the mean-squared error and cross-entropy loss, respectively.

Image Processing. Received images, sampled from the dataset during training or directly from the game during evaluation, are first resized to the required image size of the respective visual encoder (see Table 1)[2]. If image augmentation is used for an encoder, images

[1] Code for our experiments can be found at https://github.com/microsoft/imitation_learning_in_modern_video_games.
[2] For end-to-end encoders, we resize images using linear interpolation. For pre-trained encoders, we use bicubic interpolation to be consistent with the processing pipeline used during training.

Table 1: Overview of visual encoders considered in this study including the training category, image sizes, parameter counts and the size of computed embeddings. For pre-trained encoders we only report the size of visual encoder used to embed images (and do not include the parameters of other model components such as language embedding networks of CLIP models).

Category	Model	Image size	Parameters	Embedding size
End-to-end	Impala ResNet	128×128	98K	7200
	Custom ResNet	128×128	585K	1024
	Custom ResNet	256×256	586K	1024
	ViT Tiny	224×224	5.5M	192
	Custom ViT	128×128	8.8M	512
	Custom ViT	256×256	8.9M	512
Language contrastive pre-trained	CLIP ResNet50	224×224	38M	1024
	CLIP ViT-B/16	224×224	86M	512
	CLIP ViT-L/14	224×224	303M	768
Self-supervised pre-trained	DINOv2 ViT-S/14	224×224	21M	384
	DINOv2 ViT-B/14	224×224	86M	768
	DINOv2 ViT-L/14	224×224	303M	1024
Classification supervised pre-trained	FocalNet Large FL4	224×224	205M	1536
	FocalNet XLarge FL4	224×224	364M	2048
	FocalNet Huge FL4	224×224	683M	2816
Reconstruction pre-trained	Stable Diffusion 2.1 VAE	256×256	34M	4096

are augmented after resizing using the same augmentations applied by Baker et al. [1]. Finally, image colour values are normalised. Figure 1 illustrates the architecture of agents with an image o_t being processed by the visual encoder to obtain an embedding z_t that is given to the policy to obtain actions a_t.

Policy network. For all experiments, the policy architecture consists of a MLP projecting the embedding z_t of the visual encoder to a dimension of 512 before being fed through a two-layered LSTM [13] with hidden dimensions of 512. The LSTM processes the projected embedding and a hidden state h_{t-1} which encodes the history of embeddings during a sequence (obtained as a sampled sequence during training or online evaluation). The output of the LSTM is then projected to as many dimensions as there are actions in the task using a MLP with one hidden layer of 512 dimensions. At each intermediate layer, the ReLU activation function is applied.

End-to-end visual encoders. For visual encoders trained end-to-end with the BC loss, we consider three ResNet [12] and three vision transformer (ViT) [7] architectures. For ResNets, we evaluate the Impala [8] architecture as a commonly used visual encoder for decision-making agents. However, it outputs large embeddings for 128×128 images so we also evaluate two alternative ResNet architectures based on ConvNeXt [20] designed for 128×128 and 256×256 images, respectively. For ViTs, we evaluate the commonly used tiny model architecture proposed by Steiner et al. [33] which outputs fairly small embeddings. For comparison, we also evaluate two ViT architectures with slightly larger models that output slightly larger embeddings. See Section A.1 for full details on all end-to-end visual encoder architectures.[3]

[3]All appendices are available online at https://arxiv.org/abs/2312.02312.

Pre-trained visual encoders. We consider four paradigms of pre-trained visual encoders with representative models being evaluated in our experiments: OpenAI's CLIP [27] as language contrastive pre-trained encoders, DINOv2 [23] as self-supervised pre-trained encoders with self-distillation objectives between a teacher and student network, FocalNet [40] trained on ImageNet21K classification as supervised pre-trained encoders, and a variational autoencoder (VAE) [18] from stable diffusion [29] as reconstruction pre-trained encoder. These visual encoders have already been trained on large amounts of real-world images. During all our experiments, we freeze these encoders and only use them to obtain embeddings of images without any fine-tuning or further training. See Section A.2 for details on the evaluated models.

Training details. For each network update, we sample 32 random sequences of 100 consecutive image-action pairs within the dataset. Before each training step, the hidden state and cell state of the LSTMs in the policy are reset and the mean BC loss is computed across all sequences with the hidden state accumulating across the 100 samples within a sequence. The Adam optimiser [17] is used with decoupled weight decay [21] of 0.01 and a learning rate of $3 \cdot 10^{-4}$. To stabilise training, gradients are normalised at 1.0 and we use half precision for all training. In Minecraft Dungeons, we train each model for 1 million gradient updates. In Minecraft and CS:GO, models are trained for 500,000 gradient updates.

4 VIDEO GAME TASKS

We train and evaluate BC models with all visual encoders in three different games, Minecraft, CS:GO, and Minecraft Dungeons illustrated in Figure 2. Table 2 provides a summary of the tasks and datasets in each of the games. We highlight that (1) all games represent complex and popular video games, (2) the games cover varying

(a) Minecraft Dungeons **(b) Minecraft** **(c) Counter-Strike**

Figure 2: Representative screenshots of all games studied in this paper.

Table 2: Summary of evaluation video games, the size of the human demonstrations dataset, the perspective (and resolution) of input images, and the action space given by the number of continuous (cont.) and discrete binary (bin.) actions.

Game	Task	Dataset	Images	Actions
Minecraft Dungeons	Arch Haven	8h	Isometric (1280 × 720)	4 cont., 11 bin.
Minecraft	Treechop	40min	First-person (640 × 360)	2 cont., 8 bin.
Counter Strike	Clean Aim Train	45min	First-person (280 × 150)	2 cont., 1 bin.

image perspectives (isometric and first-person) and varying degrees of realism (as shown in Figure 2), (3) all games require agents to take long sequences of actions with Minecraft, CS:GO, and Minecraft Dungeons requiring up to 1,200, 4,800, and 3,000 actions, respectively, per rollout, and (4) all games have complex combinatorial action spaces including multiple continuous actions (mouse or joystick movement) and discrete binary actions (button presses) with up to fifteen actions in Minecraft Dungeons (as shown in Table 2). Below, we provide a summary of the key features of each of the considered tasks. For full details about each game, including the task, action space, dataset, and online evaluation, we refer to Section B.

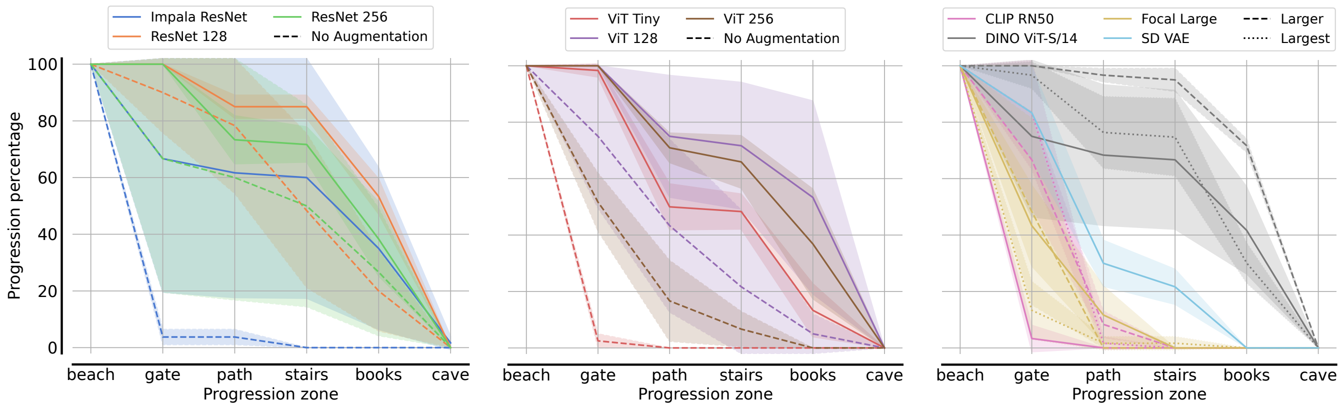
In Minecraft, we evaluate in the established MineRL "Treechop" task [11] and train on filtered human gameplay demonstrations from the VPT dataset [1]. We consider any rollout a success if the agent chops a tree within 1 minute of gameplay. Collecting a log from a chopped tree is the first step to craft many of the items in Minecraft, and has been previously used to benchmark reinforcement learning algorithms [11]. To succeed at this task, the agent needs to navigate a rich and diverse visual environment and coordinate its movement, camera movement, and "interacting" action required to chop a tree. Images in Minecraft features highly stylised visuals in the first-person perspective.

In CS:GO, we evaluate in the "Clean Aim Train" task and use an existing dataset of human expert demonstrations [26]. The agent is located in a fixed position and controls the camera to aim and the trigger to shoot. To succeed at the task, the agent needs to look around, identify enemies moving towards it and shoot them. We measure the kills per minute during rollouts as a performance metric. Images in CS:GO are in first-person and have a more realistic style compared to the other two games, so this benchmark tests if our findings generalise across video games with distinct visual styles. We further highlight that images in CS:GO appear more akin to real-world images most pre-trained visual encoders are trained on, but also exhibit few colours with little contrast between objects of interest and the grey background.

Minecraft Dungeons constitutes a novel benchmark for imitation learning which has not been evaluated in before. In this game, we train agents on demonstrations of human gameplay in the "Arch Haven" level that requires precise navigation through a level with multiple combat scenarios against randomised enemies. In particular the stochastic combat of the game represents a major challenge for imitation learning since the agents will encounter states not seen within the training data, and, thus, have to learn policies that generalise across similar states. Providing the policy with robust representations of the image inputs represents one approach to facilitate such generalisation. Additionally, successful rollouts in this task require the agent to take thousands of accurate steps. Images in Minecraft Dungeons feature a similar style to Minecraft but are always centred on the agent character and have an isometric perspective looking down on the character. These properties make Minecraft Dungeons a highly challenging and suitable evaluation task to study the efficacy of varying visual encoders for imitation learning. For more details on the "Arch Haven" level, including a 2D visualisation of the level, we refer to Section D.

5 EVALUATION

In our evaluation, we focus on the guiding question of how to encode images for data-efficient imitation learning in modern video games. The evaluation is structured in three experiments studying (1) which end-to-end visual encoder is most effective, (2) which pre-trained encoder is most effective, and (3) how do the best end-to-end and pre-trained visual encoders compare under reduced amounts of training data. For each experiment, we train each model with three different seeds and report aggregated training and online evaluation metrics. Lastly, we visually inspect the visual encoders with respect to the information they attend to during action selection, and present the computational cost of training a model with



(a) End-to-end ResNets **(b) End-to-end ViTs** **(c) Pre-trained encoders**

Figure 3: Online evaluation for BC agents in Minecraft Dungeons with (a) end-to-end ResNets, (b) end-to-end ViTs, and (c) pre-trained visual encoders. We visualise the mean and standard deviation, computed over three training seeds, of the percentage of rollouts progressing to objectives within the "Arch Haven" level. Results for the ViT Tiny model are only aggregated over two seeds, as one seed resulted in an invalid checkpoint.

different visual encoders. An outline of the computational resources used for training and evaluation can be found in Section F.

To further contextualise the evaluation performance of our models, Table 3 shows the online evaluation performance of VPT [1] models in Minecraft Treechop and human performance in the aim assist task of CS:GO, as reported by Pearce and Zhu [26]. VPT models represent transformers models trained on human gameplay in Minecraft. However, these models were trained using behaviour cloning with mangitudes more data and compute compared to our models. In addition to the base models of VPT, we also evaluate VPT models fine-tuned on early-game data (denoted with FT) which, among other tasks, contains significant amounts of demonstrations of players chopping trees as required by the task.

5.1 End-To-End Visual Encoders

To identify which end-to-end visual encoder is the most effective, we train all six end-to-end visual encoder architectures (listed in Table 1) with and without image augmentations using the BC loss. Figures 3a and 3b visualise the online evaluation performance for all models with end-to-end ResNet and ViT visual encoders, respectively, in Minecraft Dungeons. For Minecraft and CS:GO, online evaluation performance with end-to-end visual encoders are shown in Table 3 (left half). Results for Minecraft Dungeons and CS:GO exhibit consistent trends, with image augmentation improving online performance for almost all encoders, ResNet image encoders slightly outperforming ViT models, but not always by significant margins, and image encoders trained on 128×128 images performing similar or better than their counterparts with 256×256 image inputs. In CS:GO, the best-performing ResNet 128 +Aug as well as the following ResNet 256 +Aug models outperform all other visual encoders but the Impala ResNet + Aug model with high-variance performance by statistically significant margins (double-tailed Welch's test, $p < 0.05$), and exhibits performance comparable to a non-gamer human player. In Minecraft, we also observe that the input image size has no significant effect on the results. However, ViT 256 and ViT Tiny outperform most ResNets by statistically significant margins, and image augmentations harm online evaluation performance in several cases. These results suggest two main findings: (1) small images of 128×128 are sufficient to train agents in complex modern video games, and (2) both image augmentation and architecture choice (ResNet or ViT) have the potential to significantly improve performance but are game-specific.

5.2 Pre-Trained Visual Encoders

To identify which pre-trained visual encoders enable effective decision making in video games, we compare BC agents trained with the representations of 10 pre-trained encoders. The encoders are frozen during training.

In MineCraft (Table 3, right half) and Minecraft Dungeons (Figure 3c), we find that BC models with DINOv2 visual encoders generally outperform other models. In Minecraft, the largest DINOv2 ViT-L/14, significantly ($p < 0.05$) outperforms all but the noisiest models (Tiny ViT, ViT 128 +Aug, Stable Diffusion and CLIP ResNet 50) and reaches performance comparable to the largest VPT models despite having magnitudes less parameters and training budget. While smaller DINOv2 models appear better than FocalNet or CLIP, their results are not significantly different from ViT-B/14 and ViT-S/14 DINOv2 models. Stable diffusion VAE appears similarly effective to smaller DINOv2 models in Minecraft. Lastly, we observe that there is no clear correlation between the model size of pre-trained encoders and online performance. While larger DINOv2 models perform best in Minecraft, the same trend does not hold for CLIP and FocalNet where encoders with fewer parameters perform better. In Minecraft Dungeons, the BC models trained with DINOv2 ViT-B/14 pre-trained encoder outperforms all other models, including any end-to-end trained visual encoder. The stable diffusion encoder still outperforms FocalNet and CLIP visual encoders, but performs notably worse than all DINOv2 models.

Table 3: **Online evaluation performance in Minecraft (MC) and CS:GO, measured by the rate of chopping a single tree and kills-per-minute (KPM), respectively. The best end-to-end and pre-trained model in each group is highlighted in bold. We report mean and one standard deviation computed over three training seeds and indicate the number of valid seeds with stars (*) whenever training resulted in invalid checkpoints. For VPT and human baselines, no standard deviation is reported as only one model is available.**

Model name	MC success (%)	CS:GO KPM
Impala ResNet	4.00 ± 4.00**	6.78 ± 7.06
ResNet 128	12.67 ± 3.86	0.38 ± 0.29
ResNet 256	10.00 ± 2.45	1.18 ± 1.27
ViT Tiny	23.33 ± 4.19	0.09 ± 0.14
ViT 128	19.00 ± 2.94	2.56 ± 2.08
ViT 256	**24.33 ± 0.94**	0.24 ± 0.62
Impala ResNet +Aug	14.00 ± 0.00*	11.73 ± 8.42
ResNet 128 +Aug	10.00 ± 1.41	**17.40 ± 2.08**
ResNet 256 +Aug	6.67 ± 1.70	11.40 ± 4.93
ViT Tiny +Aug	20.00 ± 5.66	6.67 ± 2.25
ViT 128 +Aug	20.33 ± 8.06	7.29 ± 1.32
ViT 256 +Aug	13.67 ± 2.62	5.36 ± 1.74
CLIP ResNet50	19.33 ± 8.65	5.56 ± 0.84
CLIP ViT-B/16	11.33 ± 1.25	**6.02 ± 1.01**
CLIP ViT-L/14	11.33 ± 3.30	2.87 ± 1.85
DINOv2 ViT-S/14	22.33 ± 2.49	3.53 ± 1.68
DINOv2 ViT-B/14	25.33 ± 2.05	2.98 ± 1.92
DINOv2 ViT-L/14	**32.00 ± 1.63**	5.45 ± 1.34
FocalNet Large	16.00 ± 5.66	1.93 ± 1.33
FocalNet XLarge	15.33 ± 4.03	2.44 ± 0.58
FocalNet Huge	13.00 ± 1.41	1.36 ± 0.87
Stable Diffusion VAE	20.00 ± 5.89	0.87 ± 0.49
VPT 71M	54.00	–
VPT 248M	55.00	–
VPT 500M	25.00	–
VPT 248M FT	48.00	–
VPT 500M FT	33.00	–
Human (Non-gamer)	–	14.32
Human (Casual)	–	26.21
Human (Strong)	–	33.21

Interestingly, we observe different trends in CS:GO. The CLIP ResNet50 and ViT-B/16 encoders and the largest DINOv2 ViT-L/14 model outperform all other pre-trained encoders by statistically significant margins. However, while the best-performing pre-trained encoders in Minecraft Dungeons and Minecraft outperformed the best-performing end-to-end trained encoders, the opposite holds in CS:GO with pre-trained encoders performing significantly worse than the best-performing end-to-end visual encoders. This trend is consistent across all pre-trained encoders, and might occur due to the different visual style of CS:GO compared to the other evaluation games. The "Clean Aim Train" map in CS:GO is a comparably visually monotone environment with little contrast between background and target enemies, which might lead to representations of pre-trained encoders that lack important information.

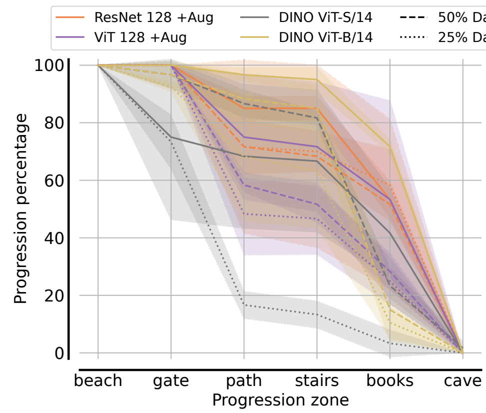


Figure 4: Online evaluation progression for the best-performing BC agents in Minecraft Dungeons with the full dataset (solid line) and subparts of the dataset.

Table 4: Online evaluation performance for the best-performing BC agents in Minecraft with the full dataset and 10% of the dataset.

Model name	Success rate (%)
ViT 256 (Full)	24.33 ± 0.94
ViT 256 (10%)	10.33 ± 1.70
ViT Tiny (Full)	23.33 ± 4.19
ViT Tiny (10%)	16.50 ± 1.50
DINOv2 ViT-L/14 (Full)	32.00 ± 1.63
DINOv2 ViT-L/14 (10%)	15.00 ± 2.16
DINOv2 ViT-B/14 (Full)	25.33 ± 2.05
DINOv2 ViT-B/14 (10%)	17.00 ± 1.41

Overall, these results suggest that the efficacy of pre-trained visual encoder is game-specific, but DINOv2 represent a strong starting point for BC agents in visually complex video games. These results complement recent findings in robotics [6] that suggest that DINO [5] can yield high-quality universal representations for data-efficient imitation learning.

5.3 How Much Data Do You Need?

A significant advantage of utilising pre-trained visual encoders is their independence from additional training, potentially resulting in more reliable performance with limited data. In contrast, visual encoders specifically trained for a particular task may be less generalisable but have the potential to outperform general-purpose pre-trained encoders. To test this hypothesis, we examine how the top-performing end-to-end and pre-trained visual encoders (based on online evaluation performance) compare when trained on reduced datasets in Minecraft and Minecraft Dungeons.

In Minecraft, we select the DINOv2 ViT-B/14 and ViT-L/14 models, and the ViT 256 and ViT Tiny models as best-performing pre-trained and end-to-end encoders and train them on a reduced dataset. The reduced dataset contains only 10% (~ 3.5 minutes,

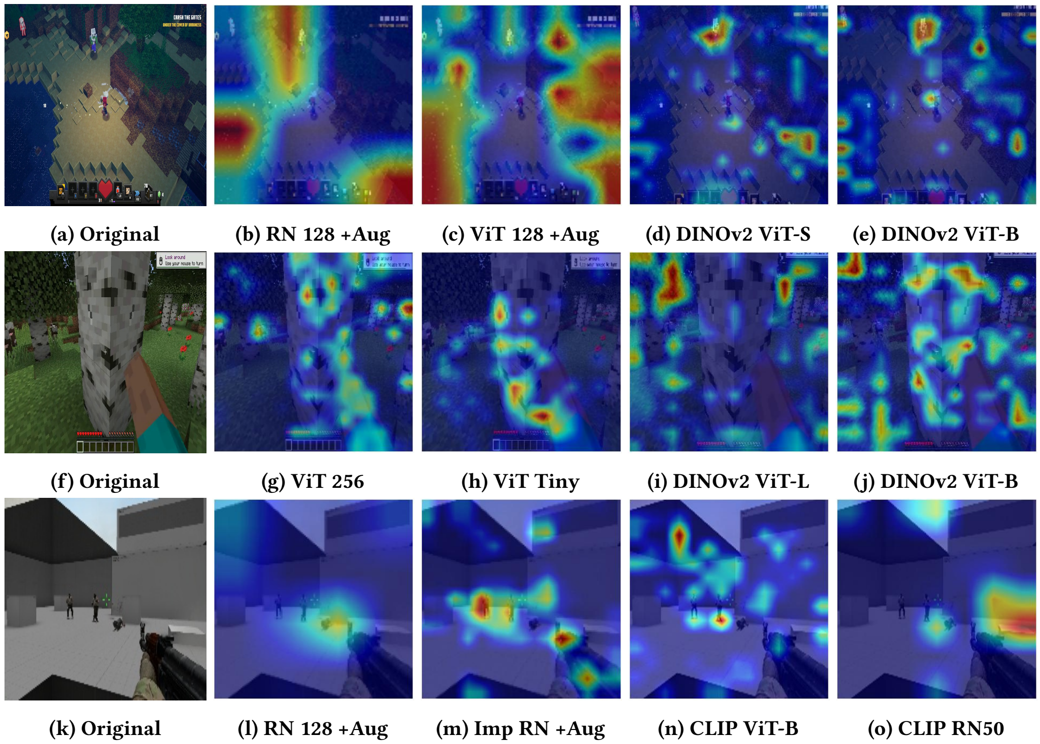


Figure 5: Grad-CAM visualisation of the activation of the best-performing visual encoders for Minecraft Dungeons (top), Minecraft (middle), and CS:GO (bottom) with action logits of a BC policy serving as targets. Red areas represent the parts of the image the visual encoders focus on the most.

14 demonstrations, 4269 total steps of data) of the original gameplay data sampled uniformly at random. Table 4 shows the rollout success rate of all models trained on the full and 10% datasets. For the reduced dataset, the success rate of all models drops by half, but they still achieve reasonable success rate and perform better than some encoders trained on the full dataset. This is surprising considering how little data these models have been trained on and how visually diverse the environment in Minecraft can be across different rollouts. Perhaps surprisingly and contrary to our initial hypothesis, models with pre-trained visual encoders are similarly affected by the reduction in training data than the end-to-end visual encoders. This finding suggests that both pre-trained and end-to-end trained encoders can work reasonably well with less than 5 minutes of high-quality demonstration data.

In Minecraft Dungeons, we select the DINOv2 ViT-S/14, ViT-B/14 models, as well as the ResNet and ViT architectures on 128×128 images with image augmentation as the best-performing pre-trained and end-to-end trained encoders, respectively. We generate two reduced datasets with 50% ($\sim$ 4 hours) and 25% ($\sim$ 2 hours) of the training data by sampling trajectories uniformly at random. Each of the selected models is then trained on the 50% and 25% training datasets for 500 and 250 thousand gradient updates, respectively. Figure 4 shows the online evaluation performance of all models. As expected, we can see that the performance of all models gradually deteriorates as the training data is reduced. For pre-trained models, the larger DINOv2 ViT-B/14 outperforms the smaller ViT-S/14 when dealing with smaller datasets. Regarding end-to-end trained models, the ViT model's performance declines more rapidly with smaller data quantities compared to the ResNet. However, similarly to results in Minecraft, both end-to-end trained visual encoders yield performance comparable to pre-trained models in lower data regimes. These findings might indicate that the rollout performance of models in this low-data regime is mostly bottlenecked by the policy rather than the visual encoder.

5.4 Grad-Cam Inspection of Visual Encoders

To understand what information is captured by the trained visual encoders, we use gradient-weighted class activation mappings (Grad-CAM) [31] to inspect each visual encoder. We visualise the Grad-CAM activations of visual encoders for images in all three games with action logits of trained BC policies serving as the targets. The resulting heatmap visualisations on top of game images can be interpreted as which parts of the image are most relevant for the visual encoder during action selection.

Figure 5 shows the Grad-CAM activations for the best-performing visual encoders in all three games. In Minecraft, most visual encoders tend to focus on parts indicative of nearby terrain, wood, and the progress of chopping a tree, which aligns with the task objective. Similarly, in CS:GO, we observe that many visual encoders focus on areas around enemies and close to edges of obstacles behind which enemies might hide and appear from. However, these correlations are less clear for most of the pre-trained encoders. In

Table 5: Training cost in training time (minutes) and VRAM (GB) usage for 2,500 training steps on Minecraft Treechop training dataset for end-to-end trained (left) and pre-trained (right) visual encoders. Numbers are the average across three runs, measured on a machine with a single A100 GPU (80GB VRAM). Runs marked with † ran out of VRAM.

Encoder	Time (min)	VRAM (GB)
Impala ResNet	17.84	21.37
ResNet 128	10.58	5.74
ResNet 256	19.72	14.30
ViT Tiny	†	†
ViT 128	16.38	21.20
ViT 256	†	†
Impala ResNet + Aug	23.23	20.01
ResNet 128 + Aug	15.73	17.97
ResNet 256 + Aug	†	†
ViT Tiny + Aug	†	†
ViT 128 + Aug	21.85	26.88
ViT 256 + Aug	†	†

	No pre-computation		Pre-computed embeddings	
Encoder	Time (min)	VRAM (GB)	Time (min)	VRAM (GB)
CLIP ResNet50	43.69	30.62	4.12	1.44
CLIP ViT-B/16	62.38	29.72	3.35	1.53
CLIP ViT-L/14	227.47	49.21	4.47	2.12
DINOv2 ViT-S/14	49.12	21.36	3.90	1.27
DINOv2 ViT-B/14	99.90	41.65	3.34	1.52
DINOv2 ViT-L/14	280.48	54.38	4.58	2.47
FocalNet Large	†	†	4.43	2.19
FocalNet XLarge	†	†	4.10	2.80
FocalNet Huge	†	†	6.45	4.07
Stable Diffusion VAE	†	†	4.92	1.46

Minecraft Dungeons, many visual encoders tend to focus on image segments containing the player character and enemy units. We hypothesise that other activations might correspond to way points the models focus on to navigate through the level. These observations suggest that both end-to-end trained and pre-trained visual encoders capture relevant information for the training tasks, but we emphasise that visualisations vary greatly between images and visual encoders, which makes it difficult to draw general conclusions from these visualisations. For more details on the Grad-CAM visualisations, plots for more game screenshots in both games and all visual encoders, see Section G.

5.5 Computational Budget for Training

Pre-trained visual encoders typically represent comparably large models, leading to costly training and inference of large batches of images, as required during training with frozen pre-trained encoders. To reduce this computational cost, we pre-compute embeddings for all images in our training dataset. We note that this pre-computation is only possible because we freeze the parameters of pre-trained visual encoders during training and we have comparably small datasets. After this pre-computation, we can train the BC policy on the pre-computed embeddings without requiring the pre-trained encoder. This process significantly reduces the cost of training in both time and memory requirements compared to both training with pre-trained encoders without pre-computed embeddings and training with end-to-end visual encoders.

To quantify the computational cost of training with different visual encoders, Table 5 presents the training time and VRAM usage of training a BC agent with all visual encoders on the Minecraft dataset. For pre-trained encoders, we present both the computational cost when training with or without pre-computed embeddings. We note that pre-computing embeddings is not possible for end-to-end trained encoders since the encoder parameters change throughout training. As expected, training with pre-computed embeddings is significantly faster and requires less memory than training with the pre-trained encoder. For example for DINOv2 ViT-L/14, training time and VRAM usage is reduced by more than 98% and 95%, respectively, compared to computing embeddings during training. With pre-computed embeddings, training with pre-trained encoders is even faster and requires notably less memory than training with end-to-end visual encoders.

6 CONCLUSION

In this study, we systematically evaluated the effectiveness of imitation learning in modern video games by comparing the conventional end-to-end training of task-specific visual encoders with the use of publicly available pre-trained encoders. Our findings revealed that training visual encoders end-to-end on relatively small images can yield strong performance when using high-quality human demonstrations, even in low-data regimes of just a few hours or minutes. DINOv2, trained with self-supervised objectives on diverse data, outperformed other pre-trained visual encoders in two out of three games, indicating its generality and suitability for video games. However, the efficacy of pre-trained encoders varied across games, with all pre-trained encoders performing worse than end-to-end trained encoders in CS:GO. Lastly, we highlighted the significant reduction in computation cost when pre-computing the embeddings with pre-trained encoders. Combined with their strong performance in the majority of games, this suggests that pre-trained encoders are an accessible and strong starting point for training agents in video games.

In order to maintain focus, our study concentrated on a specific visual encoders that allowed for a range of comparisons across architectures and pre-training paradigms. Nevertheless, our study could be complemented by exploring additional supervised-trained pre-trained encoder architectures and additional scenarios within the examined or other video games. Although our study focused on settings with available training data for the evaluation task, future work could explore the potential benefits of pre-trained visual encoders when agents need to generalise across diverse levels or maps with variable visuals.

REFERENCES

[1] Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. 2022. Video pretraining (vpt): Learning to act by watching unlabeled online videos. In *Advances in Neural Information Processing Systems*.

[2] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. 2019. Dota 2 with large scale deep reinforcement learning. *ArXiv* abs/1912.06680 (2019).

[3] Shaofei Cai, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. 2023. Open-world multi-task control through goal-aware representation learning and adaptive horizon prediction. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 13734–13744.

[4] Shaofei Cai, Bowei Zhang, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. 2023. GROOT: Learning to Follow Instructions by Watching Gameplay Videos. *ArXiv* abs/2310.08235 (2023).

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging Properties in Self-Supervised Vision Transformers. In *Proceedings of the International Conference on Computer Vision*. 9650–9660.

[6] Norman Di Palo and Edward Johns. 2024. DINOBot: Robot Manipulation via Retrieval and Alignment with Vision Foundation Models. *ArXiv* abs/2402.13181 (2024).

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

[8] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. 2018. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*. PMLR, 1407–1416.

[9] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. 2022. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Advances in Neural Information Processing Systems*.

[10] Jonas Gillberg, Joakim Bergdahl, Alessandro Sestini, Andrew Eakins, and Linus Gisslen. 2023. Technical Challenges of Deploying Reinforcement Learning Agents for Game Testing in AAA Games. *ArXiv* abs/2307.11105 (2023).

[11] William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. 2019. Minerl: A large-scale dataset of minecraft demonstrations. *ArXiv* abs/1907.13440 (2019).

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*.

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[14] Mikhail Jacob, Sam Devlin, and Katja Hofmann. 2020. "It's unwieldy and it takes a lot of time" — Challenges and Opportunities for creating agents in commercial games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 16. 88–94.

[15] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. 2016. The Malmo Platform for Artificial Intelligence Experimentation.. In *Ijcai*. 4246–4247.

[16] Anssi Kanervisto, Joonas Pussinen, and Ville Hautamäki. 2020. Benchmarking end-to-end behavioural cloning on video games. In *IEEE conference on games*. IEEE.

[17] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *ArXiv* abs/1412.6980 (2014).

[18] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *ArXiv* abs/1312.6114 (2013).

[19] Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila McIlraith. 2023. STEVE-1: A Generative Model for Text-to-Behavior in Minecraft. *ArXiv* abs/2306.00937 (2023).

[20] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A convnet for the 2020s. In *IEEE/CVF conference on computer vision and pattern recognition*.

[21] Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

[22] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. 2022. R3m: A universal visual representation for robot manipulation. In *Conference on Robot Learning*.

[23] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. 2024. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research* (2024).

[24] Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. 2022. The (Un)Surprising Effectiveness of Pre-Trained Vision Models for Control. In *International Conference on Machine Learning*.

[25] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, et al. 2023. Imitating human behaviour with diffusion models. In *International Conference on Learning Representations*.

[26] Tim Pearce and Jun Zhu. 2022. Counter-strike deathmatch with large-scale behavioural cloning. In *IEEE Conference on Games*. IEEE, 104–111.

[27] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

[28] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. 2022. A generalist agent. *Transactions on Machine Learning Research* (2022).

[29] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. 2022 IEEE. In *CVF Conference on Computer Vision and Pattern Recognition*.

[30] Moritz Schneider, Robert Krug, Narunas Vaskevicius, Luigi Palmieri, and Joschka Boedecker. 2024. The Surprising Ineffectiveness of Pre-Trained Visual Representations for Model-Based Reinforcement Learning. In *Advances in Neural Information Processing Systems*.

[31] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*. 618–626.

[32] Alessandro Sestini, Joakim Bergdahl, Konrad Tollmar, Andrew D Bagdanov, and Linus Gisslén. 2022. Towards Informed Design and Validation Assistance in Computer Games Using Imitation Learning. In *Human in the Loop Learning Workshop at the Conference on Neural Information Processing Systems*.

[33] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. 2022. How to train your vit? data, augmentation, and regularization in vision transformers. *Transactions on Machine Learning Research* (2022).

[34] SIMA Team, Maria Abi Raad, Arun Ahuja, Catarina Barros, Frederic Besse, Andrew Bolt, Adrian Bolton, Bethanie Brownfield, Gavin Buttimore, Max Cant, Sarah Chakera, Stephanie C. Y. Chan, Jeff Clune, Adrian Collister, Vikki Copeman, Alex Cullum, Ishita Dasgupta, Dario de Cesare, Julia Di Trapani, Yani Donchev, Emma Dunleavy, Martin Engelcke, Ryan Faulkner, Frankie Garcia, Charles Takashi Toyin Gbadamosi, Zhitao Gong, Lucy Gonzales, Karol Gregor, Arne Olav Hallingstad, Tim Harley, Sam Haves, Felix Hill, Ed Hirst, Drew A. Hudson, Steph Hughes-Fitt, Danilo J. Rezende, Mimi Jasarevic, Laura Kampis, Rosemary Ke, Thomas Albert Keck, Junkyung Kim, Oscar Knagg, Kavya Kopparapu, Andrew Kyle Lampinen, Shane Legg, Alexander Lerchner, Marjorie Limont, Yulan Liu, Maria Loks-Thompson, Joseph Marino, Kathryn Martin Cussons, Loïc Matthey, Siobhan Mcloughlin, Piermaria Mendolicchio, Hamza Merzic, Anna Mitenkova, Alexandre Moufarek, and Valeria Oliveira. 2024. Scaling Instructable Agents Across Many Simulated Worlds. *ArXiv* abs/2404.10179 (2024).

[35] Chintan Trivedi, Konstantinos Makantasis, Antonios Liapis, and Georgios N Yannakakis. 2022. Learning Task-Independent Game State Representations from Unlabeled Images. In *2022 IEEE Conference on Games*.

[36] Chintan Trivedi, Konstantinos Makantasis, Antonios Liapis, and Georgios N Yannakakis. 2023. Towards General Game Representations: Decomposing Games Pixels into Content and Style. *ArXiv* abs/2307.11141 (2023).

[37] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.

[38] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *ArXiv* abs/2305.16291 (2023).

[39] Peter R Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, et al. 2022. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature* 602, 7896 (2022), 223–228.

[40] Jianwei Yang, Chunyuan Li, Xiyang Dai, and Jianfeng Gao. 2022. Focal modulation networks. In *Advances in Neural Information Processing Systems*.

[41] Zhecheng Yuan, Zhengrong Xue, Bo Yuan, Xueqian Wang, Yi Wu, Yang Gao, and Huazhe Xu. 2022. Pre-trained image encoder for generalizable visual reinforcement learning. In *Advances in Neural Information Processing Systems*.