ique
d other
s robotics,
un the labels
e internet-scale
semi-supervised
ine unlabeled videos.
d data we can train an
unlabeled source of online
ft – from which we can then
native human interface (mouse
havioral prior has nontrivial zero-
ed, with both imitation learning and
n tasks that are impossible to learn from
or many tasks our models exhibit human-
irst to report computer agents that can craft
oficient humans upwards of 20 minutes (24,000
play to accomplish.

s demonstrated the efficacy of pretraining large and general foundation
rnet-scale datasets for use in downstream tasks in natural language[1–4] and
For sequential decision domains (e.g. robotics, game playing, and computer
nts must repeatedly act within an environment, a wealth of data also exists on the
, most of this data is in the form of *unlabeled* video (i.e. without the actions taken
ne), making it much less straightforward to train a behavioral prior in these domains
in e.g. natural language. In a few rare settings, such as Chess, Go, and StarCraft, there

---

his was a large effort by a dedicated team. Each author made huge contributions on many fronts over long
ne periods. All members were full time on the project for over six months. BB, IA, PZ, and JC were on the
original VPT project team and were thus involved for even longer (over a year). Aside from those original team
members, author order is random. It was also randomized between IA and PZ.

[†] OpenAI

[‡] University of British Columbia

s
ngs,
within
causal BC
distribution
le, which does
ation bottlenecks
am tasks with either

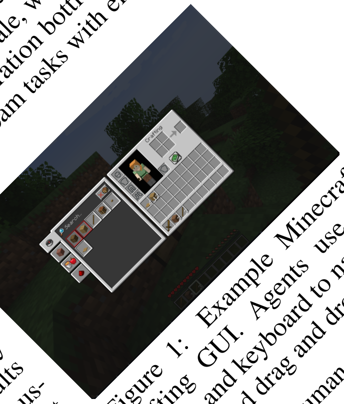


Figure 1: Example Minecraft crafting GUI. Agents use the mouse and keyboard to navigate menus and drag and drop items.

lts
er us-
nd (c) it
s a research
y difficult ex-
ne native human
st accurately model
domain shift between
e data collection easier by allowing our human contractors to
and (3) eliminate the need to hand-engineer a custom interface
vironment. This choice means that our models play at 20 frames
se and keyboard interface to interact with human GUIs for crafting,
ding dragging items to specific slots or navigating the recipe book with
). Compared to prior work in Minecraft that uses a lower frame rate and
attacking macros,[30,32–34] using the native human interface drastically increases
xploration difficulty, making most simple tasks near impossible with RL from
simple task of gathering a single wooden log while already facing a tree takes 60
ack actions with the human interface, meaning the chance for a naive random policy to
$_{2}^{60}$. While this paper shows results in Minecraft only, the VPT method is general and
applied to any domain.

ection 4 we show that the VPT foundation model has nontrivial zero-shot performance, accom-
ishing tasks impossible to learn with RL alone, such as crafting planks and crafting tables (tasks
requiring a human proficient in Minecraft a median of 50 seconds or ∼970 consecutive actions).
Through fine-tuning with behavioral cloning to smaller datasets that target more specific behavior
distributions, our agent is able to push even further into the technology tree, crafting stone tools

f
roll
vations
earning is
trajectories
deo games.[10,44]

not work; however,
ations.[22] For instance,
cy to exhibit behaviors
opose to first learn a latent
nt actions to real actions with
motion-capture methods to track
se waypoints. Similarly, Behbahani
however, they construct waypoints that
ls. Pathak et al.[49] and Nair et al.[50] train
e current state towards expert-provided goal
. Most similar to our own work, Torabi et al.[24]
l (IDM),[51] which aims to uncover the underlying
past and future timesteps, e.g. $p_{\text{IDM}}(a_t|o_t, o_{t+1})$, and
ectories of observations labeled with the IDM. Data to
the BC model in the target environment such that both
any point in training if there are sequences in the dataset that
ures that the BC model perform those or similar sequences in
correctly label them. Therefore, if the BC model does not explore
w down learning. In order to avoid this potential issue we opted for a
we first train an IDM on a small number of labeled trajectories collected
(they play the game as would normally as we record their keypresses and
ecause human contractors reach most relevant parts of the state space, we can
hroughout BC training.

ost previous work in semi-supervised imitation learning, we experiment in the much
x and open-ended environment of Minecraft. Minecraft is a voxel-based 3D video
, due its popularity and wide variety of mechanics, has attracted a vast amount of RL
n.[27,28,30–34,52–60] A large body of work focuses on small, custom-made Minecraft worlds
tasks such as navigation,[53,60] block placing,[54,55] instruction following,[58,59] combat,[56] and
ers.[28,31,57] Work operating in the massive, randomly generated environments of Minecraft itself
has included hill climbing,[52] automated curriculum learning[30] and, most closely related to the RL
experiments presented in Sec. 4.4, diamond mining.[27,32–34] However, to the best of our knowledge,
there is no published work that operates in the full, unmodified human action space, which includes
drag-and-drop inventory management and item crafting.

ly,
deed,
ll show
. This IDM
harder task of
etails.

g the web for related
ts, such as a video feed
from platforms other than
modes, e.g. in Minecraft we
ust gather or craft all their items.
s from survival mode, and call all
el, and enough training compute, a BC
ly still perform well in a clean Minecraft
ute efficiency, we choose to filter out unclean
h clean and unclean segments). We do this by
g a small dataset (8800) of images sampled from
unclean (Appendix A.2).

ndation model with standard behavioral cloning, i.e. mini-
actions predicted by the IDM on clean data. For a particular

$${}_{\theta}(a_t|o_1,\ldots,o_t), \text{ where } a_t \sim p_{\text{IDM}}(a_t|o_1,\ldots,o_t,\ldots,o_T) \tag{1}$$

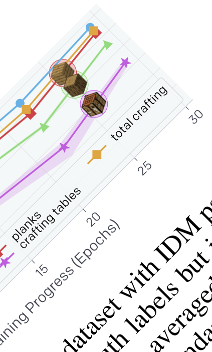
lowing sections, this model exhibits nontrivial zero-shot behavior and can be
imitation learning and RL to perform even more complex skills.

s

erformance of the Inverse Dynamics Model

e IDM architecture is comprised primarily of a temporal convolution layer, a ResNet[62] image
processing stack, and residual unmasked attention layers, from which the IDM simultaneously
predicts keypresses and mouse movements (see Appendix D for IDM architecture and training
details). A key hypothesis behind our work is that IDMs can be trained with a relatively small amount
of labeled data. While more data improves both mouse movement and keypress predictions, our best

.

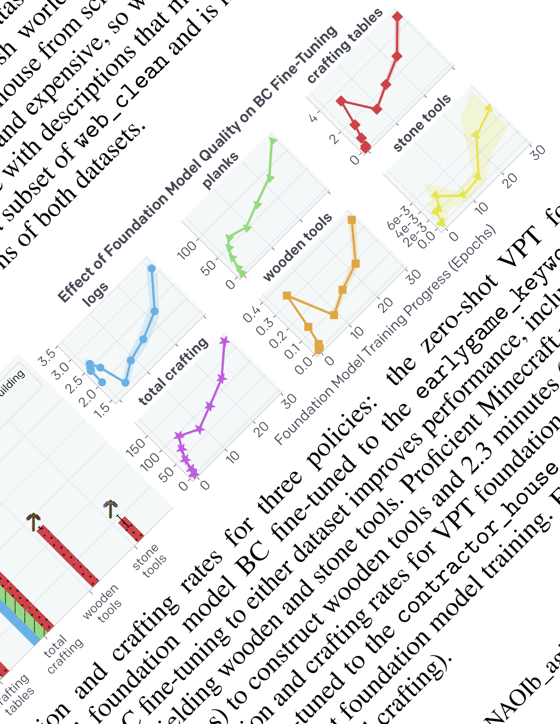


internet dataset with IDM pseudo-
has ground-truth labels but is out-of-
ollected per episode averaged over 2500
poch, shaded with the standard error of the
el, or sand (all materials that can be gathered
ing trees for three seconds, a difficult feat for an
anks can be crafted from logs, and crafting tables
n-game crafting GUIs, and proficient humans take a
ions) to make a crafting table.

r learned by a behavioral cloning policy trained on an extremely
labeled with our IDM. To collect the unlabeled internet dataset,
ble videos of Minecraft play with search terms such as “minecraft
e searches resulted in $\sim$270k hours of video, which we filtered down to
lding an *unlabeled* dataset of $\sim$70k hours, which we refer to as `web_clean`
er details on data scraping and filtering). We then generated pseudo-labels
n our best IDM (Section 3) and then trained the VPT foundation model with
g. Preliminary experiments suggested that our model could benefit from 30 epochs
d that a 0.5 billion parameter model was required to stay in the efficient learning
r that training duration (Appendix H), which took $\sim$9 days on 720 V100 GPUs.

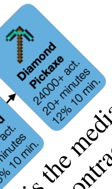
uate our models by measuring validation loss (Fig. 4, left) and rolling them out in the
craft environment. Unless otherwise noted, in all environment evaluations we spawn agents in a
ndard survival mode game where they play for 60 minutes, i.e. 72000 consecutive actions, and we
plot the mean and shade the standard error of the mean for various game statistics such as crafting
and collection rates (Fig. 4, right). The VPT foundation model quickly learns to chop down trees
to collect logs, a task we found near impossible for an RL agent to achieve with the native human
interface (Sec. 4.4). It also learns to craft those logs into wooden planks and then use those planks

ble across a
narrower task
ic datasets.[1] The
o-shot performance;
ology tree. As a case
model's ability to collect
asets targeted at Minecraft
sh world. In the first dataset,
house from scratch using primarily
and expensive, so we also construct a
with descriptions that match keywords
subset of `web_clean` and is labeled with
ns of both datasets.



ollection and crafting rates for three policies: the zero-shot VPT foun-
d the VPT foundation model BC fine-tuned to the `earlygame_keyword` or
ouse datasets. BC fine-tuning to either dataset improves performance, including (for
`tor_house` dataset) yielding wooden and stone tools. Proficient Minecraft players take
of 1.2 minutes (1390 actions) to construct wooden tools and 2.3 minutes (2790 actions)
ruct stone tools. (**Right**) Collection and crafting rates for VPT foundation model snapshots
ghout training *after* they are BC fine-tuned to the `contractor_house` dataset. In general,
afting-related behaviors increase throughout foundation model training. Fig. 4 defines the other
task terms (logs, planks, crafting tables, and total crafting).

(iv) Sample videos: https://www.youtube.com/playlist?list=PLNAOIb_agjf3U3rSvG_BCWqJ869NdBhcP



tem is the median
ntage of contractors
axe is unknown (except
of 20-minute episodes.

ging goal of obtaining a diamond
world. Doing so involves acquiring
ls like mining, inventory management,
ing a furnace, and mining at the lowest
(Fig. 6). Adding to the difficulty, progress
, or dying. Obtaining a diamond pickaxe more
utes (24,000 actions).

he sequence, with lower rewards for items that have to
tems near the end of the sequence. Agents are optimized
ithm for ~1.3 million episodes (roughly 1.4×10^{10} frames).
endix G.1 for reward function and RL training details. Due to
ments use a ~ 248 million parameter VPT model (Appendix H).

ng with RL is catastrophic forgetting[65,66] because previously learned
value is realized. For instance, while our VPT foundation model never
e of behaviors required to smelt iron zero-shot, it *did* train on examples of
rnaces. It therefore may have some latent ability to smelt iron once the many
o have been performed. To combat the catastrophic forgetting of latent skills
continually improve exploration throughout RL fine-tuning, we add an auxiliary
er (KL) divergence loss between the RL model and the frozen pretrained policy.[10]

om a randomly initialized policy fails to achieve almost *any* reward, underscoring how
exploration challenge the diamond pickaxe task is for RL in the native human action space
a). The model never learns to reliably collect logs, typically the first of many steps to obtaining
iamond pickaxe (Fig. 7b). RL fine-tuning from the VPT foundation model does substantially
better (Fig. 7a), learning everything up to mining iron ore and crafting furnaces. (Fig. 7c). However,
this agent fails at smelting an iron ingot, the next item required to get further into the tech tree, likely

---

(v) Sample Videos: https://www.youtube.com/playlist?list=PLNAOIb_agjf2yDSs4AqcoyPv4z_eWUiKm

s
e to
el) all
episodes.

s too low, even

tion Model to the
en fine-tuning with RL
er than first fine-tuning to
.2). The three-phase training
earning extremely difficult tasks:
ability on collecting diamonds, and
or comparison, human players given
tems in 57%, 15%, and 12% of episodes,
afting iron pickaxes and mining diamonds.
reliability in 15 minutes[32,33] but always with a
. To the best of our knowledge, **we are the first to**
amond pickaxe. Qualitatively, the model developed
icient mining patterns, cave exploration, returning to
es, and advanced techniques like using wooden pickaxes

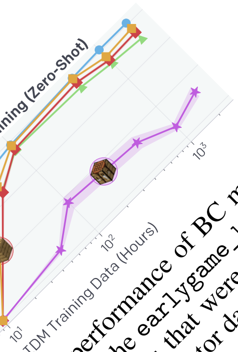
ce of the KL loss to the pretrained model during RL fine-tuning.
obtains only items early in the sequence (logs, planks, sticks, and
eward (Fig. 7a). This failure to progress further into the sequence is
tial skills of chopping logs and crafting planks are being learned with RL,
afting a wooden pickaxe are lost due to catastrophic forgetting.

g Properties of the Foundation Model

n we validate a core hypothesis behind this work: that it is far more effective to use
ontractor data to train an IDM within the VPT method than it is to directly train a BC
ion model from that same small contractor dataset. If we could cheaply collect a labeled
actor dataset of a similar order of magnitude as `web_clean`, then this would not be important;
owever, collecting that scale of data would have cost millions of dollars. Figure 8 compares
foundation models trained on increasing orders of magnitude of data from 1 hour up to the full ~70k
`web_clean` dataset. Foundation models trained up to and including 1k hours are trained on the IDM

(vi)Videos found at https://www.youtube.com/playlist?list=PLNAOIb_agjf3e_UKweM5pQUSfTw8r-Wfc

he
tools



Zero-shot performance of BC models
om scratch on the `earlygame_keyword`
t labeled with IDMs that were trained on
easing amounts of contractor data.

er which there are few to no gains and differences are
are only crafted after 50 or more hours of IDM data, and
e in all previous experiments we use our best IDM trained
ggest we could reduce that number to as low as 100 hours.

usion

is paper help pave the path to utilizing the wealth of unlabeled data on the
sion domains. Compared to generative video modeling or contrastive methods
d representational priors, VPT offers the exciting possibility of directly *learning*
aining and using these learned behavioral priors as extremely effective exploration
VPT could even be a better general representation learning method even when the
n task is not learning to act in that domain—for example, fine-tuning to explain what is
g in a video—because arguably the most important information in any given scene would be
t in features trained to correctly predict the distribution over future human actions. We leave
intriguing direction to future work.

Future work could improve results with more data (we estimate we could collect >1M hours) and larger, better-tuned models. Furthermore, all the models in this work condition on past observations only; we cannot ask the model to perform specific tasks. Appendix I presents preliminary experiments on conditioning our models on closed captions (text transcripts of speech in videos), showing they

ariwal,
nodels are
901, 2020.

t: Pre-training of
t *arXiv:1810.04805*,

qi Chen, Omer Levy, Mike
stly optimized bert pretraining

ee, Sharan Narang, Michael Matena,
imits of transfer learning with a unified
10683, 2019.

athan, Kaiming He, Manohar Paluri, Yixuan Li,
aaten. Exploring the limits of weakly supervised
pean conference on computer vision (ECCV), pages

ris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal,
Pamela Mishkin, Jack Clark, et al. Learning transferable visual
e supervision. In *International Conference on Machine Learning*,
, 2021.

rew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von
rnstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the
d risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

ai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transform-
R, abs/2106.04560, 2021. URL `https://arxiv.org/abs/2106.04560`.

d Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driess-
e, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

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

al

y random

lune. First return,

Rachita Chhaparia, Alistair
antoro, et al. A data-driven
v:2202.08137, 2022.

andez, and Percy Liang. World of
In Doina Precup and Yee Whye Teh,
ence on Machine Learning, volume 70 of
3135–3144. PMLR, 06–11 Aug 2017. URL
`117a.html`.

rithms for inverse reinforcement learning. In *Icml*,
learning from

Peter Stone. Recent advances in imitation learning from
v:1905.13566, 2019.

mon. Generative adversarial imitation learning. *Advances in neural*
ystems, 29, 2016.

tt Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv*
305.01954, 2018.

, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation:
o imitate behaviors from raw video via context translation. In *2018 IEEE International*
ence on Robotics and Automation (ICRA), pages 1118–1125. IEEE, 2018.

winfinite Staff. Most played games in 2021, ranked by peak concur-
rent players. *Twinfinite*. URL `https://twinfinite.net/2021/12/`
`most-played-games-in-2020-ranked-by-peak-concurrent-players/`.

[27] William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*, 2019.

nd
from

Juewu-mc:
arXiv preprint

network. *Advances in*

s? *Trends in cognitive sciences,*

d Brett Browning. A survey of robot
ous systems, 57(5):469–483, 2009.

Elyan, and Chrisina Jayne. Imitation learning:
ng Surveys (CSUR), 50(2):1–35, 2017.

zier, and Donald Michie. Learning to fly. In *Machine*
–393. Elsevier, 1992.

i, Dan C Cireşan, Fang-Lin He, Juan P Rodríguez, Flavio
hristian Forster, Jürgen Schmidhuber, Gianni Di Caro, et al. A
o visual perception of forest trails for mobile robots. *IEEE Robotics*
1(2):661–667, 2015.

avide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon
D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning
g cars. *arXiv preprint arXiv:1604.07316*, 2016.

devilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy.
-end driving via conditional imitation learning. In *2018 IEEE international conference*
obotics and automation (ICRA), pages 4693–4700. IEEE, 2018.

Rémi Coulom. Computing “elo ratings” of move patterns in the game of go. *ICGA journal*, 30
(4):198–208, 2007.

[44] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

gey
lation.

rning inverse
s, number CONF,

, and Robert E Schapire.
ex domains. *arXiv preprint*

el L Littman. Deep reinforcement
preprint arXiv:1902.04257, 2019.

e, and Richard Socher. Keeping your dis-
ancing shaped rewards. *Advances in Neural*

ning with model learning and monte carlo tree search
3.08456, 2018.

an, and Shim-Young Lee. Fighting zombies in minecraft with
Technical report, Technical report, Technical report, Stanford

Xiong, and Richard Socher. Hierarchical and interpretable skill acquisi-
nforcement learning. *arXiv preprint arXiv:1712.07294*, 2017.

atinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization
sk deep reinforcement learning. In *International Conference on Machine Learning*,
1–2670. PMLR, 2017.

gxiang Shi, Yue Feng, and Aldo Lipani. Learning to execute or ask clarification questions.
Xiv preprint arXiv:2204.08373, 2022.

0] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum
learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

[61] Robert George Douglas Steel, James Hiram Torrie, et al. Principles and procedures of statistics.
Principles and procedures of statistics., 1960.

n.
y of

tchell. On
eedings of the
610–623, 2021.

el, Bertrand Thirion,
t Dubourg, et al. Scikit-
g research, 12:2825–2830,

eit, Llion Jones, Aidan N Gomez,
need. *Advances in neural information*

Hinton. Layer normalization. *arXiv preprint*

B Orr, and Klaus-Robert Müller. Efficient backprop.
, pages 9–48. Springer, 2012.

Adam: A method for stochastic optimization. *arXiv preprint*

, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan,
g Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas
, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy,
Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-
ep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-
, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*,
4–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/`
`pytorch-an-imperative-style-high-performance-deep-learning-library`.

.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and
Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-
convex optimization. *Advances in neural information processing systems*, 27, 2014.

[75] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

on ap-
. PMLR,

s, Charlie Deck,
athieu, et al. Open-
107.12808, 2021.

er, Jacob Andreas, Edward
rvey of reinforcement learn-
enty-Eighth International Joint
9–6317. International Joint Confer-
doi: 10.24963/ijcai.2019/880. URL

nson, Arun Ahuja, Arthur Brussee, Federico
eorgiev, Alex Goldin, Tim Harley, et al. Creating
ion and self-supervised learning. *arXiv preprint*

, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical
n with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

matic speech recognition, volume 1. Springer, 2016.

punct, May 25 2021. URL `https://github.com/Felflare/rpunct`.
.

ntan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek,
, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. Text and code embeddings
ive pre-training. *arXiv preprint arXiv:2201.10005*, 2022.

owledgements

e thank the following people for helpful discussions and support: Bob McGrew, Ken Stanley,
Joel Lehman, Ilya Sutskever, Wojciech Zaremba, Ingmar Kanitscheider, David Farhi, Glenn Powell,
Jonathan Gordon, and the OpenAI supercomputing team, especially Christian Gibson, Ben Chess,
and Christopher Berner.

ies
r beginners
orial series
new world
al a new beginning
vival episodio 1
survival епизод 1
t survival 1. bölüm
e a new minecraft survival world

rch terms used for generating the initial web dataset.

metadata available, we perform an additional step of metadata-based filtering
s that do not fit our target distribution. In this step, we look for a list of blacklist
video title and description and reject videos that contain these terms. The blacklist
use are: {ps3, ps4, ps5, xbox 360, playstation, timelapse, multiplayer, minecraft pe,
ion, skyblock, realistic minecraft, how to install, how to download, realmcraft, animation}.
cess yielded us ∼270k hours of unlabeled data, which we filter down to only a “clean” subset
scribed in the next section.

A.2 Training a Model to Filter out Unclean Video Segments

We restrict the scope of this work to the Minecraft Survival game mode and therefore limit our training dataset to clips that are obtained from this mode that are relatively free from visual artifacts.

e image
provided in

necraft. Everything
arked as `None of the`
.

an item hotbar at the bottom of the screen.
be classified as `None of the Above`.

e - `No Artifacts`: These images will be clean screenshots
l mode gameplay without any noticeable artifacts.
l `Mode - with Artifacts`: These images will be valid survival
out with some added artifacts. Typical artifacts may include image
brand), text annotations, a picture-in-picture of the player, etc.
e `Above`: Use this category when the image is not a valid Minecraft survival
It may be a non-Minecraft frame or from a different game mode. In non-survival
odes such as the creative mode, the health/hunger bars will be missing from the
e, the item hotbar may or may not be still present.

we spent $319.96 on human labeling experiments on mTurk, of which $159.98 was directly
o workers. The remaining amount was spent towards Amazon platform fees. The workers
eived $0.01 per labeled image, at an hourly compensation of $7.20 (based on an estimated labeling
me of 5 seconds/image – in our internal sample run of the same task, we found the average labeling
time to be <3 seconds).

Since we perform rigorous keyword and metadata based filtering of videos (as described in A.1) from
which we served sample images to be labeled, serving offensive content to workers was extremely

k

vival `Mode - No Artifacts`.
ode `- with Artifacts - Image`
ght) Sample image for Class 3: `None`
alth and armor bars, indicating that it was

d in the previous section, we trained a classifier to extract
rom the `Minecraft Survival Mode - No Artifacts`
es, we obtain embeddings for each image using the RN50x64
esNet-based CLIP model that is scaled up to have approximately
0. We then train a Support Vector Machine (SVM) using the RBF
fier. We use the Scikit-learn[68] SVM implementation with the parameter
e 2.

assifier to frames of raw video sequences at a rate of 3 frames/second. We filter
ist of at least 80% "clean" frames at this stage (Classes `Minecraft Survival`
rtifacts and `None of the Above` are both considered not clean). From this set,
edian filter (with a kernel size of 7) to the labels and segment videos by splitting the
ments that are at least 5s in duration. The result of this is our final `web_clean` dataset.

`early_game` Dataset

The `early_game` dataset is a ~3000 hour subset of `web_clean` targeted at "early game" Minecraft behavior, i.e. instances where players start in a fresh world with no items. We obtain the metadata text that accompanies the videos in `web_clean` and determine whether any of the following regular expressions match:

cords their actions and game video
Reborn (github.com/Hexeption/MCP-
nvironment is as close as possible to
uations (Appendix C), we use the same
ava app that runs in a window mode, with
o 0 (the "gloomy" setting in Minecraft), which
d of view, GUI scale) are fixed to the values used
tly prevented users from changing graphics settings.
all keyboard key presses and continuous (as opposed to
step (or “tick”) the frame buffer used to display the game
written into a video file. In-game actions are recorded in a
ere each line is a JSON-formatted string). All recordings are
er each 5 minute segment of contractor game play the recorder
eo file, the JSONL file with actions, as well as a Minecraft state file.
cannot corrupt each other’s data, we provided every contractor with an
s well as with credentials giving write access only to that bucket. Credentials
-adjective-noun names (e.g. grumpy-amethyst-chipmunk), generated with the
ython package to ensure contractor anonymity when we publish the data.

actor Contract

dited contractors by posting the following offer on the UpWork freelancing platform.

> “We are collecting data for training AI models in Minecraft. You’ll need to install java, download the modified version of Minecraft (that collects and uploads your play data), and play Minecraft survival mode! Paid per hour of gameplay. Prior experience in Minecraft not necessary. We do not collect any data that is unrelated to Minecraft from your computer.”

or stone

ft a diamond
s used to obtain
k (and the subtasks
goal.

ude personally identifiable
necraft's open-world property
ontent (e.g. by using Minecraft
pot from which the message would
o so in the contractor videos that we
the internet of people playing Minecraft,
also potentially learn it, although we expect
t be likely to reproduce it.

ven the video, any labelled data is appropriate for IDM
ral gameplay as well as the `treechop` task data described
nted to a total of 1962 hours. Due to collecting datasets like
tages of the project, they were not included in IDM training.

se contains about 420 hours of data. We asked contractors to build a basic
, using only basic dirt, wood, and sand, blocks. Each trajectory starts in a newly
d a timer forcibly ends a trajectory after a 20 minute time limit. For this task, many
se to begin their trajectories by crafting basic tools and building blocks, specifically
on for the first 2 minutes to be spent crafting a wooden pickaxe and then mining stone
rtment of stone tools before gathering more building blocks and beginning to create their
re.

C Minecraft environment details

Our Minecraft training environment is a hybrid between MineRL[27] and the MCP-Reborn (github.com/Hexeption/MCP-Reborn) Minecraft modding package. Unlike the regular Minecraft

ar
tions.
agent's
g but not as
nutes for BC
ater for too long
o not terminate the
game, the agent drops
pawning spot in the same
an remember the fact that it

s from the Minecraft game that a human
ys like the hotbar, health indicators, and the
he attack or “use” actions. The field of view is
efault. GUI scale (a parameter controlling the size
s is set to 2 (which is not a Minecraft default, but is
rendering resolution is 640x360, which is downsampled
odels. We empirically found 128x128 to be the smallest
ements are still discernible, and then chose that to minimize
ame GUI is open, we additionally render an image of a mouse
position to match what a human player's operating system does (Fig.

includes almost all actions directly available to human players, such as keypresses,
ents, and clicks. The specific binary actions we include are shown in Table 3.
ence between the human action space and our agent's is that we disallow typing arbitrary
which is only useful for entering text into the search bar of the crafting recipe book. Humans
ither do that or browse the recipe book with the mouse, the latter of which our agent can still
. However, because we do allow the agent to press letters that are also shortcuts for actions (e.g.
outside of the GUI, the "W" key triggers the `forward` action) agents are able to press a few keys
within the GUI (W, A, S, D, E, Q) that produce letters if the recipe book search bar is selected. We
have not seen agents attempt to search the recipe book with these letters. Instead, our agents navigate
the recipe book with the mouse or craft by dragging items around the crafting window.

player
s or place

tems the player
presses ctrl-Q then
GUI, the same thing
mouse is hovering over.
ne in a given hotbar cell.

minecraft.fandom.com/wiki/

ur action space also includes mouse movements.
are not open, mouse X and Y actions change the
GUI is open, camera actions move the mouse cursor.
ve the mouse or camera relative to the current position,
nt position).

quires fine-grained mouse movements to achieve tasks such as
g and navigating the world can be achieved with coarser mouse
oth with the same action space, we implemented mouse movements
ith foveated binning along each axis (Fig. 13), which in preliminary
mprove crafting performance.

namics Model Training Details

Architecture

M model has approximately 0.5 billion trainable weights. The input to the IDM is 128
ecutive image frames (128 frames of video), each of which has dimensions $128 \times 128 \times 3$. The
M is tasked with predicting the action at each frame. All image pixel values are first divided
by 255.0 such that they lie within the range $[0, 1]$. The first layer of the IDM is a 3-D convolution
with 128 learnable filters with a temporal kernel width of 5 and spatial kernel widths of 1. This
convolution is non-causal, meaning that embeddings at time index t are functions of pixel values at
times $t-2, t-1, t, t+1$, and $t+2$. We found this layer to be extremely important in IDM training

g
ault

e IDM Architecture.

a ResNet[62] image processing network. In this
s shared between neighboring frames; however,
emporal convolution, some temporal information
rocessing network is comprised of three subsequent
Each stack is comprised of, in order, (1) an initial 3x3
dding at the embedding boundary (such that the outgoing
the incoming embedding dimension) with W output channels,
2 and padding 1 such that the embedding width and height are
Net blocks as defined in He et al.[62] with each layer also having W

stack is flattened into a 1-dimensional vector of size $2^{17} = 131072$ (one
n the video) such that at this stage there are 128 vectors of size 131072. Each
tly processed with two frame-wise dense layers with 256 output activations and
activations, respectively. The result is then fed through 4 subsequent non-causal
dual transformer[69] blocks. Each block first has an unmasked attention layer, i.e. frames
o future frames, with 32 attention heads of dimension 128 each and a surrounding residual
on that skips this layer. The embedding is then passed through a frame-wise dense layer
utput dimension 16384 and another with output dimension returning to 4096; a single residual
nection skips past this pair of frame-wise dense layers (not skipping past each layer separately,
ut skipping the pair). All dense layers have their weights tied through time, so each frame in the
video is processed with the same weights.

Finally, independent dense layer heads for each action are pulled from the final embedding – a 2
class on/off categorical parameterized with a softmax for each available key as well as a 11-way

results
mitigated
e data points

k than behavioral cloning
sly predict all 128 actions for
mes at the end of the video clip
s reason, we apply the IDM over a
y use the pseudo-label prediction for
e IDM prediction at the boundary of the
es of a full video.

ing

ure is the same as the IDM architecture described in Appendix
ecture so that it is causal (i.e. cannot see the future when making
architecture does not have the initial non-causal convolution the
d completely). Furthermore, the residual transformer layers are now
dard in language modeling) and we do Transformer-XL-style[76] training
to keys and values from past batches within the same video. We also use a
relative attention position embedding.

on Filtering

common action humans take is the null action (no keypresses or mouse movements), which
s for 35% of all actions they take. Among other reasons, a player may take the null action to
or something in the game to finish, to pause between actions, or to take a break to grab a glass
water. Early on in the project we found that the BC model would take a much larger fraction than
35% of null actions, often upwards of 95%. In order to prevent this behavior we removed frames with
null actions from the dataset. We compare a few different treatments: we filter nulls if there have
been 1, 3, or 21 frames of consecutive null actions, and include a treatment that does not perform
any null filtering. Null action filtering generally helps, increasing all crafting rates (Figure 15 left).

g
more
iments

d be independently on
moved. This could cause
en state, humans either with
y (b) move left and drop their
bility to each of the 4 constituent
d independently. See Appendix C.2

actions; however, the full joint distribution
ensions discretized into 11 bins each would
ions. This is far too large for many reasons, e.g.
mension of 4096 would need to be mapped to each
5.2×10^{11} parameters for this final layer alone. In
ttons in Minecraft have no effect when simultaneously
ove forward and backward at the same time, they remain in
utually exclusive actions. Furthermore, the inventory button is
mouse movement.

Mutually Exclusive Actions

`forward`, `back`
`left`, `right`
`sprint`, `sneak`
`hotbar.[1-9]`

g the joint action space to reflect these mutually exclusive combinations still results in a
n space when combined with the discretized mouse movements, i.e. $3^3 \times 10 \times 2^4 \times 11^2 + 1 \approx$
^{5}. This calculation results from 3^3 for the 3 sets of 2 mutually exclusive keys above where
g neither in the set is an option, $\times 10$ for the 9 hotbar keys or no hotbar keypress, $\times 2^4$ for the
maining binary 4 keys: `use`, `drop`, `attack`, and `jump`, $\times 11^2$ for mouse movements, and finally
$+1$ for the `inventory` button which is mutually exclusive with all other actions. $\sim 5.2 \times 10^5$ is still
quite large so we chose to implement a hierarchical binary action for camera being moved or not. If
this action is on, then there is a secondary discrete action head with 121 classes (the joint distribution
of mouse movements because each discretized mouse direction has 11 bins) that determines where

eing
sted in

training

on model training, except we either use a focused
ribed in A.3) with pseudo labels, or contractor data
) with ground-truth labels. The hyperparameters used
able 5. We used 16 A100 GPUs for about 6 hours when
et, and 16 A100 GPUs for about 2 days when fine-tuning

Hyperparameter	**Value**
Learning rate	0.000181
Weight decay	0.039428
Epochs	2
Batch size	16

Table 5: Hyperparameters for behavior cloning fine-tuning

einforcement Learning Fine-Tuning

.1 Reinforcement Learning Fine-Tuning Training Details

RL experiments were performed with the phasic policy gradient (PPG) algorithm,[64] an RL algorithm based on the proximal policy optimization (PPO) algorithm[77] that increases sample efficiency by performing additional passes over the collected data to optimize the value function as well as an

s
alue
shared
n requires
or the three
Table 6. In our

BC pre-training, they
mented as a single, fully
ned model (Appendix D.1).
hile the weights of the regular
prevent destructive updates early
ction. To prevent the value-function
tude of the reward, we normalize the
ng by the standard deviation, which are
erage.

he pretrained network when RL fine-tuning, we
e RL model and the frozen pretrained policy.[10] This

$${}_{t} = \rho\,\mathrm{KL}(\pi_{pt}, \pi_\theta) \qquad (2)$$

ed, π_{pt} is the frozen pretrained policy, $\mathrm{KL}(\pi_{pt}, \pi_\theta)$ is the
n the policy being trained and the pretrained policy, and ρ is a
ive to other losses.

s, this KL divergence loss replaces the common entropy maximization
o RL experiments to encourage exploration.[79,80] The idea behind entropy
hen all actions appear to have equal value, such as when the agent has not
t reward, it should maximize its entropy to increase the chance that it discovers
blindly exploring by maximizing entropy is effective when the state and action
ciently small or the reward is sufficiently dense, but becomes infeasible when the
on spaces are large and rewards are sparse, which is the case in the diamond-pickaxe
ad of blindly exploring through uniform-random actions, we assume that the pretrained
as an action distribution that is much more likely to take sequences of actions that lead
erestingly new states, and thus, in states where the agent assigns equal value to each of its
ions, it should mimic the action-distribution of the pretrained policy instead of a uniform-random
action distribution. In experiments with a randomly initialized policy we do include the entropy
maximization loss with a coefficient of 0.01, which has been an effective setting in other Minecraft
work.[30] Empirically, we found that a high coefficient ρ for this KL divergence loss would prevent
the agent from properly optimizing the reward function while a low coefficient ρ was ineffective at

s,
best
ntially
lect logs.
g without a
y too much in a
ptimized yet, and
cond, when running
divergence decay, but
previous work.[30]

ting catastrophic forgetting. As such,
ixed factor after each iteration (Table 6).
guaranteeing that the policy can eventually
rent its behavior has to be to do so relative to

quantities of each item that a human player might
xe, and we reward the model for gathering up to that
mates by iterating over the technology tree backward from
irements for each item to the reward function (e.g. first we
rd function, then we added the 3 diamonds and 2 sticks required
hen we added the 1 iron pickaxe required for mining diamonds,
al and torches to the reward function, with coal being useful as fuel
crafting torches while the torches themselves improve visibility and
wning. Finally, we reward the model for bringing additional logs (5 logs are
ns in the reward function, but we reward up to 8 logs), which can be used as
crafting table or sticks if the agent runs out. In practice the agent rarely collects
s, places the torches, or uses coal as fuel when smelting, but the reward function
uman expectations on what would be useful to execute this task, rather than designed
an RL model behaves after training. Finally, to encourage the agent to keep mining
s and crafting diamond pickaxes after it has crafted its first diamond pickaxe, we did not put
on the number of diamonds or diamond pickaxes that would be rewarded.

e rewards for the different items are separated into 4 tiers, roughly depending on how late a player
would usually get the relevant item. The first tier consists of all wooden and stone items and has a
base reward of 1, the second tier consists of all items requiring coal with a base reward of 2, the third
tier consists of all items requiring iron with a base reward of 4, and the final tier is diamond with a
base reward of 8. Thus items later in the sequence of items towards a diamond pickaxe generally

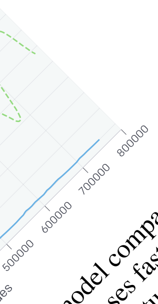
to
table,
ve divide
he purpose
y for diamond
g rewarded). For
being in the iron tier
e is $4/3$. The quantity

ed, the reward function is still
he fact that it can take thousands
red all the necessary prerequisites
nd a diamond after crafting an iron
ost efficient method for getting one item
example, a good strategy for the agent to
ew seconds to pick up) its crafting table after
mmediate access to a crafting table as soon as it
astest way to get a reward for gathering cobblestone
ooden pickaxe, while leaving the crafting table behind.
ering cobblestone makes it more difficult to learn to craft

6 days (144 hours) on 80 GPUs (for policy optimization) and
ng rollouts from Minecraft). In this time the algorithm performed
ations and collected roughly 1.4 million Minecraft episodes consisting
total of 16.8 billion frames.

Learning Fine-Tuning Additional Data

s that are helpful for understanding the main results of the RL fine-tuning experiments
in this section. First, we show the items-over-training figure when RL fine-tuning from
game model without a KL loss (Fig. 16). When training without a KL loss, the model
ns to obtain the four items that the early-game model is capable of getting zero-shot, which
gs, planks, sticks, and crafting tables. Second, we present preliminary experiments in which
directly compare RL fine-tuning from the house-building model and RL fine-tuning from the
early-game model (Fig. 17). These experiments differ from the main experiments in that, for both
treatments shown here, the KL loss coefficient was set to 0.4, the learning rate was set to 6×10^{-5},
and the reward for each item was $1/quantity$ for all items (i.e. items closer to the diamond pickaxe
did not have an increased reward). While RL fine-tuning from the house-building model initially



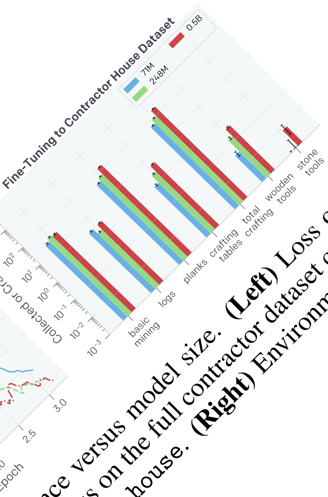
y-game model compared to
initially increases faster when
y-game model eventually obtains
game model has a higher likelihood
l was chosen for future RL fine-tuning

sing model size led to models staying in the efficient
ere we compare the 0.5B model described in Section 4.2 to
l. Both of these models are trained for 15 epochs as compared
ained for. These models have the same architecture as the 0.5B
M parameter model has 1/2 the width and each layer in the 71M
. The 71M model was trained with an initial learning rate of 0.001586,
ght decay of 0.044506. The 248M model had an initial learning rate of
40, and weight decay of 0.051376.

w validation loss on `web_clean` with IDM pseudo-labels, loss on the contractor
n the IDM with ground truth labels collected during contractor play, and zero-shot
erformance for the 71M, 248M, and 0.5B models. While larger models have better
oss on `web_clean`, these results do not tell the clear story that the 0.5B model is better
maller counterparts. The 71M model has the lowest contractor dataset loss while having the
`web_clean` loss, and it also has the best zero-shot environment performance. In fact, we see
the 71M model even had non-zero wooden tool crafting (Fig. 18 bottom left). The 248M model
lso appears to be better at crafting than the 0.5B, and also has lower contractor dataset loss.

While the zero-shot results suggest smaller models are better, fine-tuning tells another story. When fine-tuning to `contractor_house`, model size rank ordering reverses and now the 0.5B model performs best both in validation loss (Fig. 19 left) and in environment performance (Fig. 19 right)

a.
ctor
or data.
Zero-shot
odel (bottom



formance versus model size. **(Left)** Loss on the
Middle) Loss on the full contractor dataset collected
m `contractor_house`. **(Right)** Environment rollout

el and then the 71M model. Environment model rollouts are performed
gine that we use to collect contractor data, which could be visually distinct
om the web. It is plausible that the larger models overfocus on the visual
o data during pretraining since they have worse contractor data loss (Fig.18 top
s causes them to perform more poorly in the environment zero-shot. However, we
hat because the `contractor_house` dataset we fine-tune to is collected from our game
larger models that are a better overall Minecraft prior (as indicated by lower `web_clean`
on loss in Fig.18 top left) can quickly shift their low level features to perform better on data
ng from our game engine, resulting in better environment rollout performance. This hypothesis
further supported by Fig. 19 (middle) showing loss on the contractor dataset collected for IDM
training, which has no overlap with `contractor_house`. After just a few steps of fine-tuning to
`contractor_house`, all models quickly improve in loss on the full IDM contractor dataset, with
larger models now performing best. While not conclusive, we believe this investigation provides
some intuition for future studies of model scaling for sequential decision making problems.

in
odel:
wooden
cally rather
). An alternate
e in Section 4.4
work[30] . However,
irst, it is flexible and
reconceive of the task to
o-shot agents like GPT, but
s on computers or in simulated
re difficult to specify via reward
doing it–more complex things like
, VPT+text could conceivably produce
n the powers of GPT to meta-learn, follow
n the form of agents that can act in virtual
imilar embodied sequential decision domains.
t we began a first step towards exploring in that

mentary from the player. This commentary is sometimes
the videos, or could be extracted post-hoc using automated
set features about 17k hours of content with associated closed

parameter VPT foundation model used in the RL-fine-tuning ex-
for the same reason: to reduce compute costs) with an additional
the subset of our data for which closed captions are available. To obtain the
irst split videos into 30 second chunks. The same text is associated with every
nk, and is made up of all the closed captions occurring within that chunk, as well
t preceding and following the chunk (if any). Because the vast majority (around
osed caption data lacks capitalization and punctuation, it is punctuated using the rpunct
e then obtain a text embedding vector of length 4,096 from the OpenAI embedding API[89] ,
processed by a randomly initialized multi-layer perceptron (MLP) with two hidden layers of
,048. The resulting activations are added for each frame to the pretrained model activations
ore the transformer layers (`pretransformerActivations += mlp(textEmbedding)`). The
odel is fine-tuned for four epochs.

Our model shows evidence of steerability. When conditioned on sentences that incite the agent to
explore (such as "I'm going to explore" and "I'm going to find water") the agent travels significantly
farther from its spawn point (Figure 20a). Additionally, we can steer the agent to preferentially collect

o stand out are shown in
atistics are measured over 5
and “water” text strings should
asks, which is what occurs. Grass
n is likely why the “seed” condition
ve to a biome with grass). The travel
he farthest point the agent reached during
of wheat seeds. The “seed” variant collects
of a steerable agent. **(c)** Collection of oak (the
ariant collects significantly more oak logs, as is to
hat the “water” variant collects less because there are
“dirt” and “dig” variants collect a large amount, and are
of “dig”) conditioned to collect dirt. It is easy to mistakenly
r trees when collecting seeds or wood, which likely explains
ollected by these variants. In all cases, the error bars are 95%
over 1,000 episodes per conditioning variant. Treatments for which
t overlap are statistically significantly different at a $p < 0.05$ level.

Variant name	**String**
dig	I’m going to dig as far as possible
dirt	I’m going to collect dirt
explore	I’m going to explore
house	I’m going to make a house
seed	I’m going to collect seeds
water	I’m going to find water
wood	I’m going to chop wood

Table 8: Strings corresponding to each conditioning variant.