Entity Multiplexing Through Activation Strength: Understanding goals in A Maze Solving Agent

Benjamin Sturgeon*
MATS, University of Cape Town

Jonathan Shock
University of Cape Town

Abstract

In this work we provide an extensive analysis into the operations of a maze solving reinforcement learning agent trained in the Procgen Heist environment. We target this model because it presented a high degree of polysemanticity due to the fact that it has to target multiple different entities to succeed. By focusing on an agent that has to target multiple similar entities we hope to answer questions about how each of these entities might be processed by the network. Our main finding is that the signals related to the targeting of different entities are encoded at different activation strengths within a single channel in the network. These "steering channels" are often highly redundant, with large numbers of channels enabling precise agent steering, but often only within narrow ranges of activation values. We also discover a paradoxical ablation effect in which removing both steering channels and navigation circuits improves entity collection rates compared to partial ablation, suggesting unexpected interference between these systems. These findings demonstrate that amplitude-based multiplexing is a fundamental strategy for encoding multiple goals in RL agents, while our counterintuitive ablation studies suggest surprising specialization and informational dependencies within the network.

1 Introduction

Understanding how deep learning models perform their tasks is currently an unsolved problem in the field of AI. This state of affairs ensures that highly reliable, controllable, and understandable models remain out of reach. This is particularly the case in the field of reinforcement learning (RL) that has been somewhat neglected by the techniques of mechanistic interpretability (MI) whose focus has largely been on Large Language Models (LLMs). With techniques from RL being applied to frontier models to a greater extent, it is our belief that lessons learned from applying MI to RL based agents can yield generalizable insights that can improve our understanding of AI agents and neural networks in general.

This research builds on the work of Mini et al. [2023] in which precise control of a maze solving agent was achieved by intervening directly on activations within the network. The current contribution begins by extending their work to an environment that involves multiple competing entities that an agent needs to reach in sequence, using the Procgen Heist environment as the target of our research.

This work focuses on a small specific model trained on an environment that has unusual properties, specifically that there are multiple entities which the model has to approach in sequence and which are similar enough in nature as to require similar capabilities in each case. The Procedurally generated nature of the training environment naturally forces the agent to learn sufficiently general representations of different entities to allow us to hope to find clear representations of these in the

*Corresponding author: bwm.sturgeon@gmail.com

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

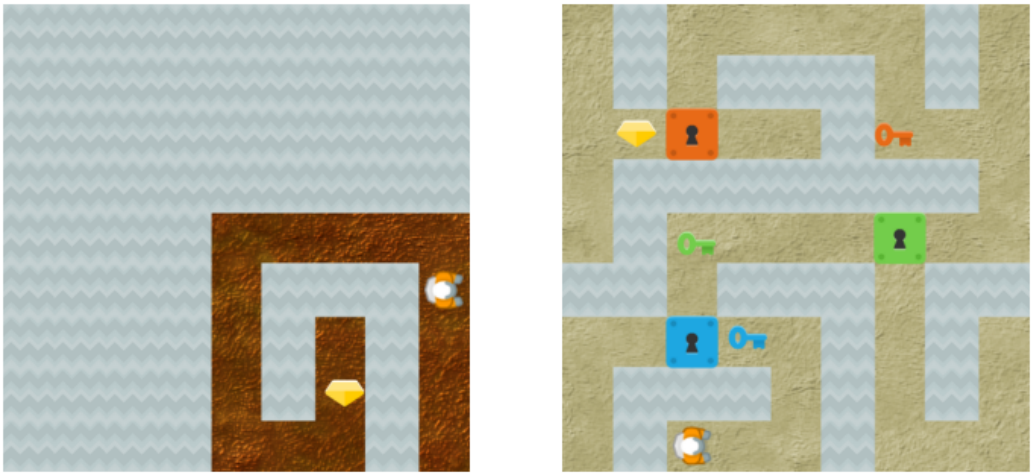

Figure 1: The Procgen Heist environment presents a procedurally generated maze where agents must collect keys and unlock corresponding doors in a specific sequence (blue, green, red) before reaching the final goal (gem). The environment varies in complexity, sometimes only having the gem present, and other times including multiple keys and locks.

weights. If the model could simply encode easily memorized heuristics then this might lead to uninterpretable and highly specific rules rather than general representations of entities.

2 Key findings

Our key findings include:

- Steering (redirecting the agent via activation manipulation) works using a single channel across multiple layers (conv3a, conv4a) across checkpoints during training.
- While some channels seemed to both track certain entities and were able to affect steering, some channels were highly efficient at tracking, but couldn't modify the location the agent would go to, indicating specialized roles in the model.
- The model was still able to navigate to different entities despite only having a single channel active.
- There exist many regions within channels in the model that were capable of successful agent re-targeting in the presence of different entities at different activation strengths, revealing a mechanism for multiplexing information within a single channel.
- Channels capable of independent navigation (can successfully lead to the agent to pick up entities even when they are the only functional channel) had an inverse relationship in their importance to overall navigation. Specifically, critical 'infrastructure channels' that have significant impact on entity pickup rates when removed form largely distinct groups from 'independent navigation' channels.
- We discovered the surprising phenomenon of models gradually exploring a variety of global minima of possible solutions to the Procgen Heist environment without degrading performance as the model changes.

3 Related Work

3.1 Mechanistic Interpretability in RL

Previous work by Mini et al. [2023] demonstrated the ability to manipulate an agent's navigation in a simple maze environment with an intervention to a single channel in the network. This serves as a foundation for our work, though we extend it to a more complex multi-objective setting. The idea of

activation steering originated in RL and was later successfully applied to LLMs Turner et al. [2024], demonstrating the potential for the transfer of techniques between very different neural network architectures and domains.

Work exploring the Procgen Coinrun environment provides a rich set of ideas for how to attribute attention from entities in the input to specific weights within the model, as well as visualizing the model weights Hilton et al. [2020]. They showed clear positive and negative attributions to specific entities according to how it might affect its ability to complete the task within the environment.

Other interpretability approaches for RL include saliency-based methods Greydanus et al. [2018], Atrey et al. [2020] which identify important input pixels but do not enable behavioral manipulation. There

More recent work also includes ambitious efforts to understand the functioning of a Video Pre-trained model (VPT) designed to play the video game Minecraft and was fine-tuned with RL methods Jucys et al. [2024]. They demonstrate the effectiveness of analyzing the attention heads in the transformer model, and demonstrate the ability to identify a case of goal misgeneralization using their methods.

3.2 Ablation Studies and Network Interpretability

Systematic ablation studies have been fundamental to understanding CNNs since Zeiler and Fergus [2013] pioneered their use to identify important model components. However, recent work has revealed that networks exhibit surprising robustness to ablations McGrath et al. [2023]. While early interpretability work searched for individual neurons encoding specific concepts, Morcos et al. [2018] demonstrated that networks achieving good generalization distribute information redundantly, with no single neuron being critical for performance.

Network Dissection Bau et al. [2017] assumes one-to-one neuron-concept mappings, but more recent work has demonstrated that neural networks are often polysemantic Elhage et al. [2022] and show that networks store multiple semantic concepts in superposition. Sparse Autoencoder research Bricken et al. [2023] successfully disentangles these polysemantic representations by training an autoencoder to separate semantic representations into a sparse overcomplete basis.

4 Methods

4.1 Motivation for the Procgen Heist Environment

The Procgen Heist environment requires the agent to collect up to 3 keys and 3 locks before reaching the final goal, a gem. The order in which the keys need to be collected is always the same (blue, green, red). The environment is procedurally generated and might generate with any combination of no keys, or 1-3 keys. The procedural generation ensures that learned representations must be general and thus robust, rather than providing memorized heuristics.

The fact that this environment uses incremental goals, where the order is essential to success and attempts to collect in the wrong order, allows us to understand the particular regime of critically discrete goal collection in a complex environment.

The environment provides sparse rewards, only giving out a score of 10 if the agent makes it all the way to the final goal and 0 otherwise. The fact that the environment provides a variety of difficulties with varying numbers of entities establishes a natural curriculum for the policy.

4.2 Model Training

4.2.1 Compressed Impala Model

We train a compressed version of the Impala model that uses 5 convolutional layers instead of 15 as used in the original Impala paper. This architecture comes from Hilton et al. [2020] where they find that this model was more interpretable while still converging to maximum environmental rewards. We include the model architecture in full in Appendix A.

For training, we employed a straightforward PPO implementation from the open-source Procgen-pfrl codebase, specifically tailored for Procgen environments. Due to computational constraints, we trained exclusively on the easy distribution of the environment.

Unlike standard Procgen Heist training which typically uses 200-500 fixed environments, we trained with unlimited procedurally generated environments. We found this was necessary for successful training with our compressed 5-layer architecture, as training collapsed when we attempted to use the standard 200-500 environments. Training on a larger variety of environments seemed likely to produce better general representations within the model, and thus more interpretable results. Unless otherwise specified, we adopted standard PPO hyperparameters: clip range 0.2, discount factor $\gamma = 0.999$, GAE parameter $\lambda = 0.95$, entropy coefficient 0.01, and value loss coefficient 0.5.

All training took place under the normal environments generated by the Procgen Heist. While we do experiments in modified environments, the agent was never trained on these, meaning that for these cases the policy was out of distribution. While this will certainly impact the behavior of the model we believe that they were similar enough to capture key behavioral dynamics of interest.

We tested model training over a number of different batch sizes and distributions. The model checkpoints used to derive our main findings were trained with standard hyperparameters: learning rate 5e-4, 64 parallel environments collecting 256 steps each (16,384 total steps per update), processed in batches of 8 over 3 epochs for 800 million steps.

The training dynamics were significantly impacted by changes to batch size, often completely collapsing training performance, but over the course of 5 runs with separate seeds with the parameters above we were able to replicate similar model dynamics in each case.

4.2.2 Sparse Autoencoder Training

To validate that our findings represent genuine mechanistic structure rather than artifacts of polysemanticity, we trained Sparse Autoencoders (SAEs) Bricken et al. [2023] on the convolutional layers following the 1×1-conv architecture from Gorton [2024].

SAEs were trained for 6M steps with an L1 warm-up schedule, achieving >99% variance explained while maintaining policy reward within 99% of baseline. We used expansion factors of 4× for all layers (e.g., conv4a: 32 channels $\rightarrow$ 128 SAE latents) with L1 coefficients ranging from 5e-4 (conv1a) to 5e-3 (conv4a). Five SAEs were trained for conv3a and conv4a to verify robustness. The resulting SAEs were then subjected to the same steering and ablation experiments as the base model layers, confirming that the amplitude-based multiplexing patterns persist in the SAEs, indicating that they reflect the network's fundamental organizational strategy rather than polysemantic confounders.

4.3 Activation Patching Experiments

When doing our experiments, we would create a customized maze setup by placing a specific entity into the environment, and then intervene on a single channel in a single layer by applying a zero mask to the channel and setting a specific region of the channel corresponding to a position on the right of the maze to a preset value. The purpose of this was to observe to what extent a channel had an influence over navigation with respect to a specific object and how this varied at different activation strengths. Note that by default the agent will always travel to the entity on the left of the maze, and we regard a successful intervention as one where the agent travels to the region corresponding to the artificial activation on the right.

In this experiment, we would sweep over a range of intervention values between 0 and a value $1.25 \times v_{\text{max}}$, where v_{max} is the maximum value that the channel would reach during typical functioning as determined by sampling from the environment during operation. This number was chosen to capture all the natural activation values of the channel, as well as slightly beyond that in order to capture instances where steering would continue to work even when out of distribution.

To establish a baseline for normal activation ranges, we calculated the 99.9th percentile of activations for each channel over 50 episodes in unmodified mazes. This threshold is used in later analysis to identify whether successful interventions operate within or outside the network's typical activation distribution.

By running this sweep, we can get a sense of what the different channels do at different intervention strengths. We kept the length of the episode to 20 steps to ensure that the agent would either be able to reach the target zone, or the actual entity, but not both. Later we refer to ranges of activation values that were successful in steering specific entities as 'intervention spans'.

For each entity-channel combination, we ran 500 trials with the activation value incrementing linearly: $v_i = i \cdot \frac{v_{\max}}{N}$, where i is the trial index (1 to N), $N = 500$ is the total number of trials, and $v_{\max}$ is the maximum activation range multiplied by 1.25.

We use a 'Q' shaped maze with the design visible in Figure 2 because it was challenging enough that the agent would need to retain its navigational abilities to reach either target, while providing a clear decision point where the agent would need to choose between pursuing the original entity or our artificial target zone.

We test a single entity at a time in this way meaning that steering that is effective with a particular value on a given channel may not be effective for redirecting the agent in the presence of another entity.

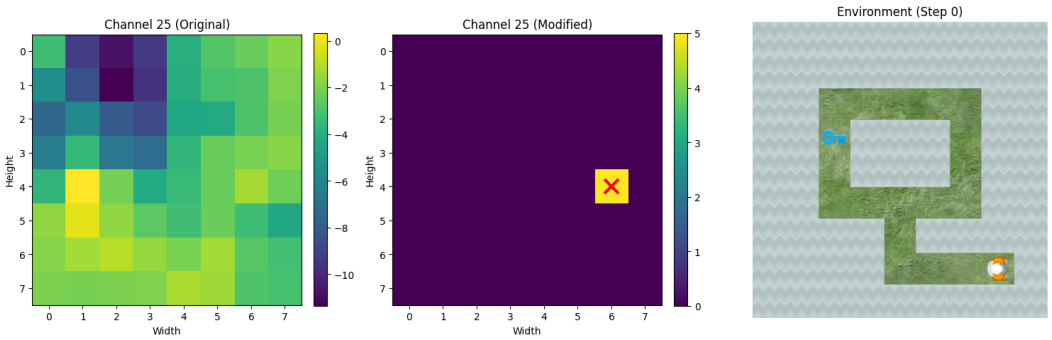

Figure 2: Example of the intervention experiment. We test channels to see the extent to which a single channel is capable of steering the agent to another location given the presence of a single entity in the environment. We consider an intervention a success if the agent enters the modified region. The maze is a 9 by 9 grid, while the conv4a channels have dimensions of 8 by 8, leading to a slightly imperfect mapping.

4.4 Channel Ablation Study

To systematically assess the role of individual channels, we developed a comprehensive ablation methodology. For each channel in the network, we ran episodes with only that channel active while zero masking all other channels. This allowed us to measure the isolated capability of each channel to support navigation to different entities in the environment. We recorded successful entity collection counts for each channel across 400 episodes, providing a quantitative measure of each channel's specialization and capability. This methodology allows us identify which channels are sufficient for basic navigation.

We use the fork shaped maze (a) seen in Figure 3 to create an environment that would allow the agent to reach all entities but would present some navigational challenge.

4.4.1 Discovering critical infrastructure channels

Complementing the previous analysis, this experiment distinguishes between channels sufficient for navigation in isolation versus those critical for the network's collective functioning. Although the above experiment reveals the degree to which individual channels can retain navigational capabilities, it is not necessarily the case that these individual channels are the most essential to successful network navigation under normal circumstances. Channels that were poor at independent navigation might be highly dependent on other channels in order to function.

To establish which channels were critical to the overall functioning of the network, we performed a similar experiment, but instead of ablating all channels but one, we ablated only one channel to observe the impact on the collection rates of different entities. We repeated this for every channel

in conv4a for n=400 episodes using maze (a) shown in Figure 3. We describe these channels as infrastructure channels in later experiments. The metric by which we ordered these channels was on how great the negative impact was on the collection rates for different entities, as channels that are more important for collecting a given entity should have a greater impact on collection rates.

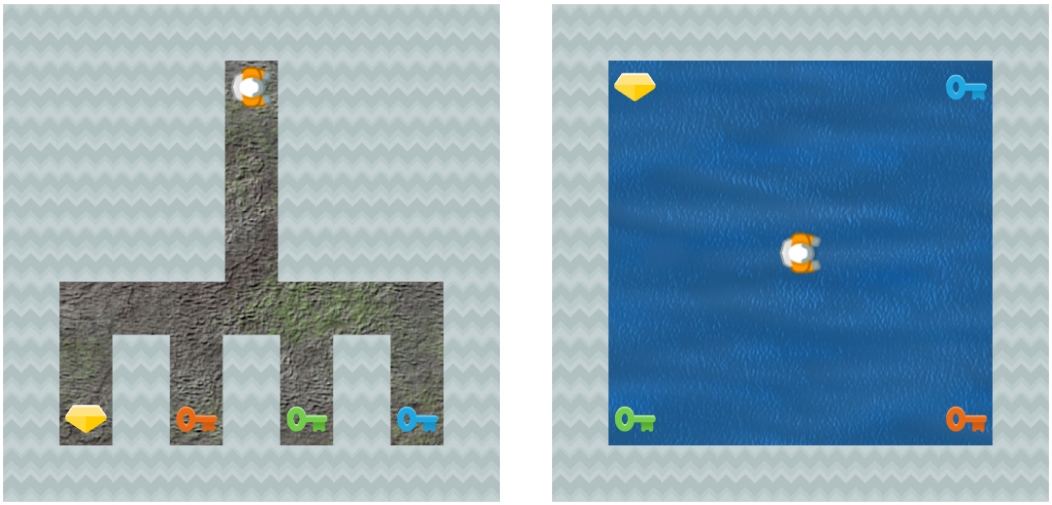

Figure 3: Maze configurations for ablation studies. (a) Fork maze tests individual channel navigation capability by requiring explicit path choices towards different entities and providing sufficient complexity that navigational abilities must be preserved. We orient the stem toward the model's inherent bias direction to minimize false positives. (b) Open maze with entities in corners tests the effect of ablating intervention spans without navigational constraints, ensuring only preferences regarding entity collection are observed.

5 Expanded ablation experiments

To derive further insights from our earlier results, we ablate regions of the network based on their success in modifying model behavior in the previous experiments. Our experiment uses a modified maze which removes all walls, and has the keys and gem randomly distributed in the corners of the maze.

We perform six targeted ablations based on our steering and navigation findings:

Ablation 1 (Intervention spans): We zero out any activations within the value ranges that successfully steered the agent. Specifically, for entity e and channel c, we set $h_{c,i,j} = 0$ whenever the activation falls within the successful steering range $\mathcal{S}_{e,c}$ identified in our intervention experiments.

Ablation 2 (Independent navigation channels): We completely ablate the top 10 channels that had the highest collection rates for a given entity in our independent navigation study, setting these channels to zero while leaving the rest unchanged.

Ablation 3 (Infrastructure channel Ablation): We ablate the top 10 infrastructure channels that, when ablated individually, caused the lowest collection rates for a given entity, setting these channels to zero while leaving the rest unchanged.

Ablation 4 (Combined): We ablate both the navigations that had the best collection rates for a given entity channels and the intervention spans for a given entity simultaneously.

Ablation 5 (Inverse): We preserve only the intervention spans that successfully steered the agent in the presence of entity e, ablating everything else. This tests whether these spans alone are sufficient for navigation.

Ablation 6 (Random): We randomly select 10 channels (equal to the number of navigation channels ablated) and completely ablate them as a baseline.

6 Results

All results presented are from checkpoint 35001 (35k update steps), well past convergence which occurred around 30k. We verified that these core findings-activation-based entity multiplexing, navigation redundancy, and the interference effects replicated across multiple checkpoints post-convergence. However, the specific channels and activation ranges shift due to the representational drift discussed in Section 6.2.

6.1 Quantitative Steering Results

We apply our incremental steering experiments across all channels in conv3a and conv4a and find a large number of channels which successfully steer the agent away from a given entity. We present our results from conv4a here, as its effective value ranges were more sparse and provided clearer results.

The most striking aspect of our results is the fact that particular value ranges worked for specific entities, illustrating how the channel seemed to signal the presence of a particular entity through the amplitude of the activation within a channel. For almost all entities shown in4, we see that single channels will modify agent behavior in the presence of all entities for all channels besides 10, 24, 25, 26, and 28 which primarily feature red key and gem steering.

The black lines and gray regions in the figure indicate regions of the channel beyond the 99.9th percentile of activations observed in 50 standard episodes, meaning that the majority of successful steering occurred outside the normal activation distribution. This means the steering results themselves should be seen as atypical of model behavior, and represent the result of a superstimulus to the model. However, the varying ranges of the activations on a per entity basis demonstrate that modified activation ranges likely play a key role in the selection of different goals in the model.

We trained SAEs on these layers and found they exhibit similar multiplexing patterns as what we discovered in the base model, confirming this is a robust organizational strategy rather than the result of polysemanticity. For additional results demonstrating the effects of steering on the SAEs, please refer to Appendix F.

6.2 Representational Drift Across Checkpoints

One observation from our experiments was that intervention positions changed completely across different checkpoints post-convergence. Despite maintaining similar performance, the network continued exploring different encoding schemes within the solution space. This was likely a result of the unlimited environments that we used during training, and is likely not typical for policies trained on fixed Procgen Heist distributions.

We find a notable checkpoint at update step 30001 where no steerable channels at all are found for any entity, though we only found 1 example of this in the 60 checkpoints we tested, suggesting that the development of steering channels is a fairly natural phenomenon in solution space. It is also possible that more such steering channels exist for that network in environmental configurations that we did not explore.

See Appendix C for detailed visualizations showing how steering behavior shifts across checkpoints 40k-60k.

6.3 Channel Ablation Studies

Our ablation analysis reveals that the ability to navigate is surprisingly redundantly encoded throughout the network. The heatmap in Figure 6 (see Appendix B) shows the results in full. We observe that all channels have some success with reaching the entities some of the time with the best performing channels achieving success rates greater than 40% for certain entities, while others have lower than 10% success rates between entities. We also see that some channels show similar performance across entities, while others are significantly better at particular entities.

6.3.1 Infrastructure Channels for Entity Targeting

We additionally investigated the impact of ablating single channels to identify which channels were most critical to the network's overall functioning. We ablated the top 10 infrastructure channels for

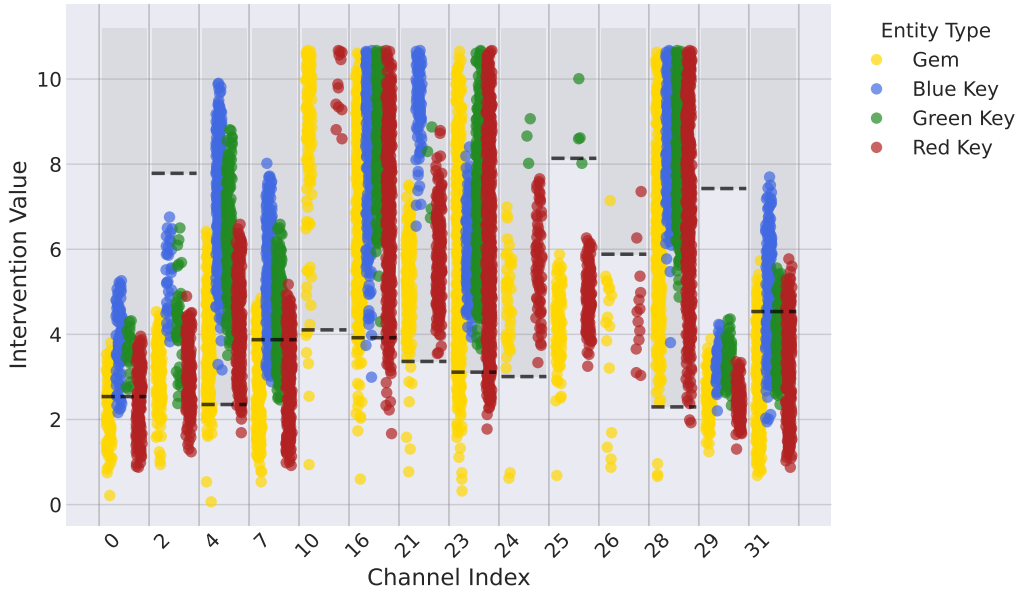

Figure 4: Steering intervention success rates for conv4a layer at checkpoint 35k. To determine the frequency with which a given channel can be used to steer the model away from a given entity, we create an artificial maze and place an artificial activation at a given point of the maze. Each colored dot represents a successful steering intervention out of the 500 runs for different entities (blue key, green key, red key, gem), while black lines indicate the 99.9th percentile of normal activations. The Y axis shows different activation values applied to the neuron in the channel, and each run uses a slightly increased increment. Intervention success is partially localized to specific activation value ranges based on entity, demonstrating amplitude-based multiplexing within individual channels.

each entity based on the results of the single channel ablation experiment and found that certain important infrastructure channels were shared between entities, but also found some specialization amongst them.

In particular, channel 25 was a critical channel which had a significant impact on collection rates for all entities in the fork maze experiments, reducing collection rates by 18% on average. We observe that ablating the channels identified as critical for gem collection had a far greater impact on the collection rates of other entities than removing their respective critical top 10 infrastructure channels. This may be the result of a greater share of the core navigational functionality being encoded in the gem infrastructure channels than in the key infrastructure channels. For detailed results see Table 3 in Appendix E.

6.3.2 Expanded ablation study

Our ablation studies reveal three core findings, the evidence for which can be seen in Table 1.

First, ablating the gem's steering channels improved collection rates from 74% to 93.2%, while ablating the key related spans reduced key collection by 47% on average. This is likely a result of the distribution of steering spans: gem-related spans appear across 31/32 channels, whereas key-related spans appear in only 14 channels. In addition, key steering spans typically appear as regions within larger gem spans. When we ablate gem-associated regions, we remove most key steering functionality along with the gem signals. The remaining channels, which still encode some weak gem targeting and are now free of interference from key signals, actually perform better, explaining the improved collection rate.

Second, we observed that intervention signals can negatively impact key collection. Specifically, ablating only the navigation channels reduced key collection rates to 45% on average; ablating the intervention spans in addition increased key collection rates to 53%. This suggests that, in the absence of navigation channels, intervention spans contribute signals that interfere with key collection. We

hypothesize that the model encodes the gem and the keys in partially distinct ways. As evidence of this, many intervention channels are activated broadly across all entities or show a stronger activation for the gem (and, to a lesser extent, the red key) rather than being uniformly distributed across all entities. By removing these conflicting signals, the agent's alternative navigation mechanisms are able to collect keys more reliably; once the keys are acquired, the task of collecting the gem is straightforward.

Third, we disovered that the channels that were most capable of independent navigation were almost completely irrelevant in terms of impacting pickup rates or different entities. When comparing the infrastructure channels and independent navigation channels we found that they shared only 1 channel in the top 10 of each type (as measured by impact on pickup rates when removed to successful independent collection rates respectively). For further discussion and data on these findings see Appendix E.1.

Table 1: Effect of targeted ablations on entity collection rates (n=400 trials per condition). Bold values indicate performance improvements or minimal degradation compared to baseline. Values shown with 95% confidence intervals.

Condition	Gem	Blue Key	Green Key	Red Key
Baseline (No Ablation)	**74.5% ± 4.3%**	98.5% ± 1.2%	95.0% ± 2.1%	80.0% ± 3.9%
Redundant Navigation Channels	45.0% ± 4.9%	69.6% ± 4.5%	55.9% ± 4.9%	43.1% ± 4.9%
Infrastructure Channels	27.3% ± 4.4%	70.2% ± 4.5%	34.5% ± 4.7%	26.5% ± 4.3%
Intervention Spans	64.8% ± 4.7%	**99.8% ± 0.4%**	**97.8% ± 1.4%**	**83.2% ± 3.7%**
Navigation + Intervention Spans	32.1% ± 4.6%	71.9% ± 4.4%	50.7% ± 4.9%	37.1% ± 4.7%
Preserve Only Intervention Spans	31.8% ± 4.6%	61.8% ± 4.8%	43.6% ± 4.9%	32.5% ± 4.6%
Random Ablation (Control)	44.1% ± 4.9%	74.9% ± 4.3%	63.1% ± 4.7%	48.9% ± 4.9%

To further investigate the distributed nature of these representations, we performed inverse ablations where we preserved only the intervention spans for each entity. This revealed a clear preference of pursuit: blue key maintained 44% performance (well above the 26% random baseline), while gem performance dropped to 16% (below the 24% random baseline). This confirms that early-game entities have more localized representations while the final goal requires whole-network context.

7 Discussion

7.1 Solution Multiplicity and Representational Drift

As shown in Section 6.2, intervention positions changed completely across different checkpoints post-convergence. The continuous drift between encoding schemes post-convergence indicates the network exists on a 'valley floor' of viable solutions, constantly reorganizing its internal representations while preserving behavioral performance. This is consistent with work showing that neural networks can occupy interconnected regions of low loss, enabling seamless transitions between solutions Draxler et al. [2019], Garipov et al. [2018]

This has meaningful implications for approaching understanding the inner workings of models. The specific channels and activation ranges we identify for steering are not fixed properties of the task but rather snapshots of a dynamic system exploring equivalent representations as possible solutions. Depending on the training regime, shifting representations could lead to brittle interpretability solutions if not taken into account.

7.2 Activation-Level Entity Encoding

Our findings regarding how the intervention spans encode multiple entities at different activation strengths reveal a sophisticated information compression mechanism. Rather than requiring separate channels for each entity type, the model has learned to use activation magnitude as an additional dimension for encoding information. This suggests that interventions on neural networks may need to consider not just which channels to modify, but also the precise activation values to use.

An explanation for why this happens might be that due to the similarity of the task of tracking each key, it proves the most efficient solution to re-purpose channels already capable of entity detection

in general, and to have them simply target each one sequentially, while indicating the nature of the given entity by varying the activation strength across channels.

8 Limitations

Our work is currently limited by the fact that we test only on a single environment, as there are not many environments that have similar qualities such as the extreme generalization encouraged by Procgen, and the staged multi-goal configuration in Heist. Future work would explore other environments which have strong discrete goal targeting dynamics.

Our training methodology differs from standard practice in using unlimited procedurally generated environments rather than the typical 200-500 set of environments one would set. While this was necessary for successful training with our compressed architecture, it may affect the generalization of our findings to models trained under standard Procgen protocols. The unlimited environment diversity likely also contribute to the representational drift we observe, as the network continually trains on new environmental distributions.

We also do not yet fully characterize the function of the steering spans, or if their steering translates to an actual unique role within the network's natural functioning. We plan to address this directly through a thorough application of probes that are trained to detect which entity is currently the next entity in order of pursuit by the network. Early work shows that the later convolutional layers do this very successfully, as well as the first fully connected layer. However, to truly understand the role of the intervention spans and navigation channels, it may be necessary to train probes on every channel and every row in the fully connected layer to determine their function.

Our approach of simply intervening in every channel with 400 different activation strengths is also an approach that would be prohibitively expensive in larger models.

9 Conclusion

Our work demonstrates that entities can be multiplexed within single channels in a neural network, and exploring these channels demonstrates an interesting structure that indicates that many channels are responsible for multiple entities, as evidenced by their effective steering in different activation ranges.

These results show that there is specialization that occurs within the network both at the level of activation amplitude to signal specific entities, and also between channels where certain channels seem to play a role of directing the heading of the network while other channels provide navigational functionality to the agent.

10 Future Work

Promising directions for future work include:

- Better characterize the precise mechanism behind goal targeting in the network.
- Explore how goal structures may encode similar multiplexing strategies in different architectures.
- Better understand the role of negative activations in network operations, as this investigation focused only on positive activation interventions.
- Do further analysis on the difference in the treatment of keys and locks within the network.
- Establish a clearer understanding of the role played by the intermediate layers of the convolutional and fully connected layers, and how spatial information is turned into strategic variables in the fully connected layers.

As RL techniques increasingly influence frontier models, understanding these fundamental organizational principles will likely provide valuable insights into building interpretable and controllable AI systems.

References

A. Atrey, K. Clary, and D. Jensen. Exploratory Not Explanatory: Counterfactual Analysis of Saliency Maps for Deep Reinforcement Learning, Feb. 2020.

D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network Dissection: Quantifying Interpretability of Deep Visual Representations, Apr. 2017.

T. Bricken, A. Templeton, J. Batson, B. Chen, A. Jermyn, T. Conerly, N. Turner, C. Anil, C. Denison, A. Askell, R. Lasenby, Y. Wu, S. Kravec, N. Schiefer, T. Maxwell, N. Joseph, Z. Hatfield-Dodds, A. Tamkin, K. Nguyen, B. McLean, J. E. Burke, T. Hume, S. Carter, T. Henighan, and C. Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

F. Draxler, K. Veschgini, M. Salmhofer, and F. A. Hamprecht. Essentially No Barriers in Neural Network Energy Landscape, Feb. 2019.

N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, R. Grosse, S. McCandlish, J. Kaplan, D. Amodei, M. Wattenberg, and C. Olah. Toy models of superposition. *Transformer Circuits Thread*, Sept. 2022.

T. Garipov, P. Izmailov, D. Podoprikhin, D. Vetrov, and A. G. Wilson. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs, Oct. 2018.

L. Gorton. The Missing Curve Detectors of InceptionV1: Applying Sparse Autoencoders to InceptionV1 Early Vision, June 2024.

S. Greydanus, A. Koul, J. Dodge, and A. Fern. Visualizing and Understanding Atari Agents. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1792–1801. PMLR, July 2018.

J. Hilton, N. Cammarata, S. Carter, G. Goh, and C. Olah. Understanding RL Vision. *Distill*, 5(11): e29, Nov. 2020. ISSN 2476-0757. doi: 10.23915/distill.00029.

K. Jucys, G. Adamopoulos, M. Hamidi, S. Milani, M. R. Samsami, A. Zholus, S. Joseph, B. Richards, I. Rish, and Ö. Şimşek. Interpretability in Action: Exploratory Analysis of VPT, a Minecraft Agent, July 2024.

T. McGrath, M. Rahtz, J. Kramar, V. Mikulik, and S. Legg. The Hydra Effect: Emergent Self-repair in Language Model Computations, July 2023.

U. Mini, P. Grietzer, M. Sharma, A. Meek, M. MacDiarmid, and A. M. Turner. Understanding and Controlling a Maze-Solving Policy Network, Oct. 2023.

A. S. Morcos, D. G. T. Barrett, N. C. Rabinowitz, and M. Botvinick. On the importance of single directions for generalization, May 2018.

A. M. Turner, L. Thiergart, G. Leech, D. Udell, J. J. Vazquez, U. Mini, and M. MacDiarmid. Steering Language Models With Activation Engineering, Oct. 2024.

M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks, Nov. 2013.

A Technical Appendices

A Model Architecture Details

ImpalaCNN architecture:

- Input: Image (H,W,C) → Normalization [0,1] → Format adaptation
- Conv Block 1: Conv(C→16, 7×7) → ReLU → LPPool(2×2, s=2)
- Conv Block 2: Conv(16→32, 5×5) → ReLU → Conv(32→32, 5×5) → ReLU → LPPool(2×2, s=2)

- Conv Block 3: Conv(32→32, 5×5) → ReLU → LPPool(2×2, s=2)
- Conv Block 4: Conv(32→32, 5×5) → ReLU → LPPool(2×2, s=2)
- Flatten → Linear(flattened→256) → ReLU
- Linear(256→512) → ReLU
- Dual heads: Policy(512→num_outputs) & Value(512→1)

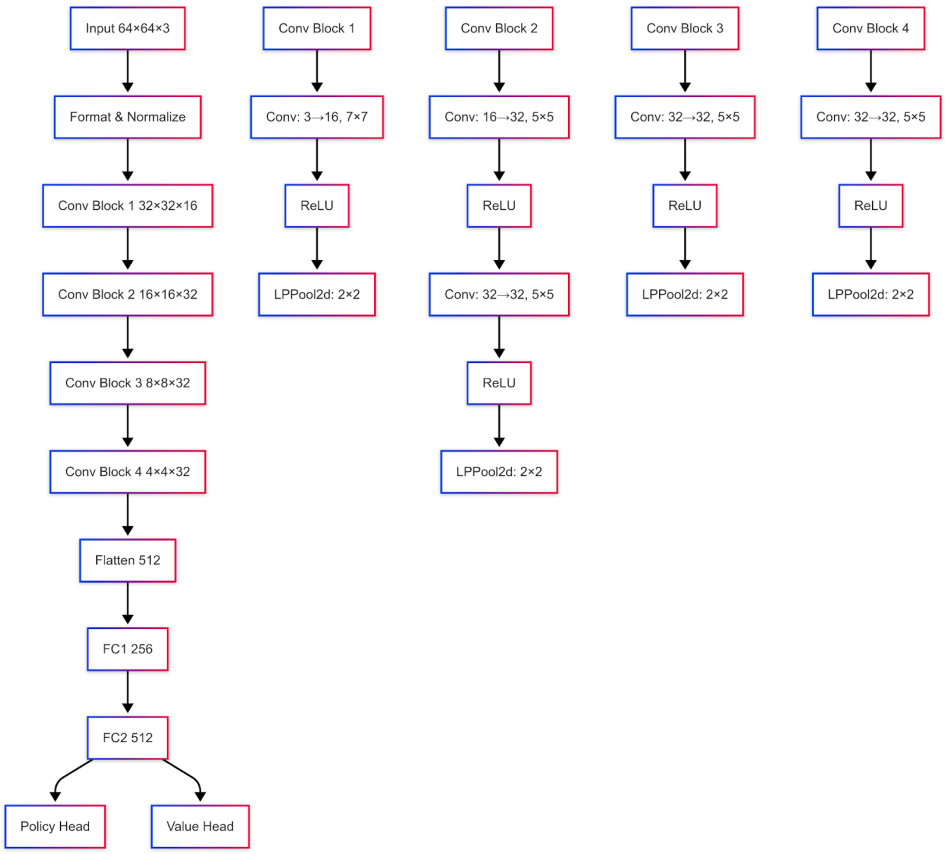

Figure 5: The convolutional neural network architecture used in our experiments. We use a modified version of the Impala CNN with 5 convolutional layers rather than the original 15 layers, which provides better interpretability without compromising performance.

B Detailed Channel Ablation Results

This section provides detailed per-channel ablation results corresponding to the heatmap shown in Figure 6. For each channel in conv4a, we measured entity collection success rates when only that channel was active (all other channels ablated to zero). Results are from 400 episodes in the fork maze environment.

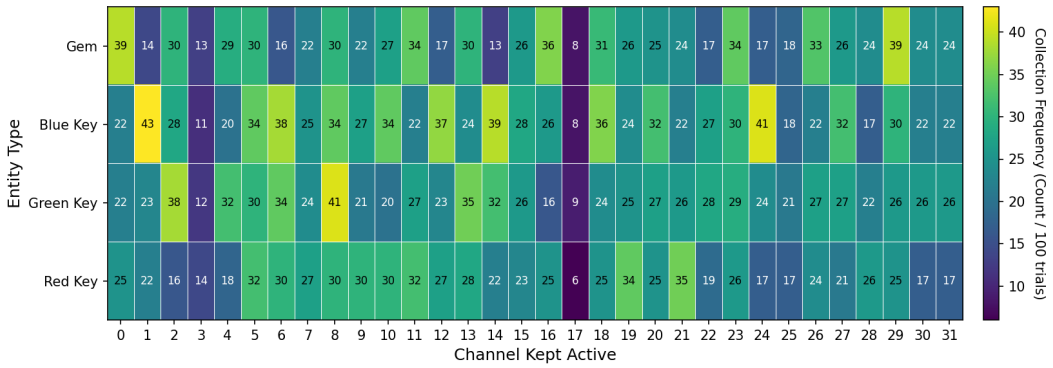

Figure 6: Heatmap showing entity collection counts when only a single channel is active (all others ablated). The vertical axis shows different entity types (gem, blue key, green key, red key), while the horizontal axis shows channel indices. Brighter colors (yellow) indicate higher collection success rates.

C Intervention spans across checkpoints

Figures 7–11 illustrate the representational drift observed across checkpoints 40k-60k. Despite stable task performance, the specific channels exhibiting steering behavior and their associated activation ranges shift dramatically between checkpoints, demonstrating that the network continuously explores equivalent solutions within a broad basin of the loss landscape.

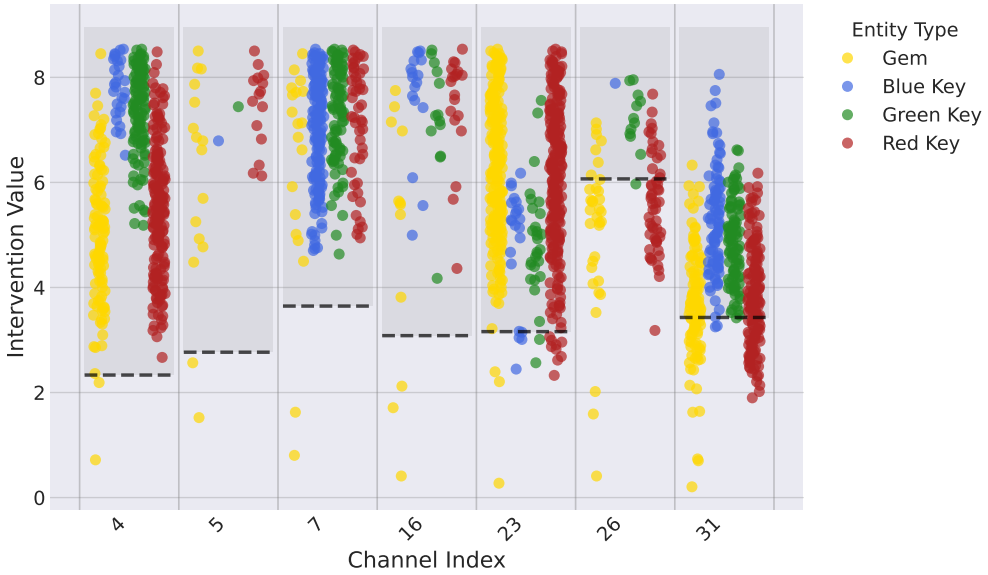

Figure 7: Steering intervention success rates for conv4a layer at checkpoint 40k. Colored dots represent successful steering interventions for different entities (blue key, green key, red key, gem), while black lines indicate the 99.9th percentile of normal activations. Each panel shows a different channel exhibiting steering behavior.

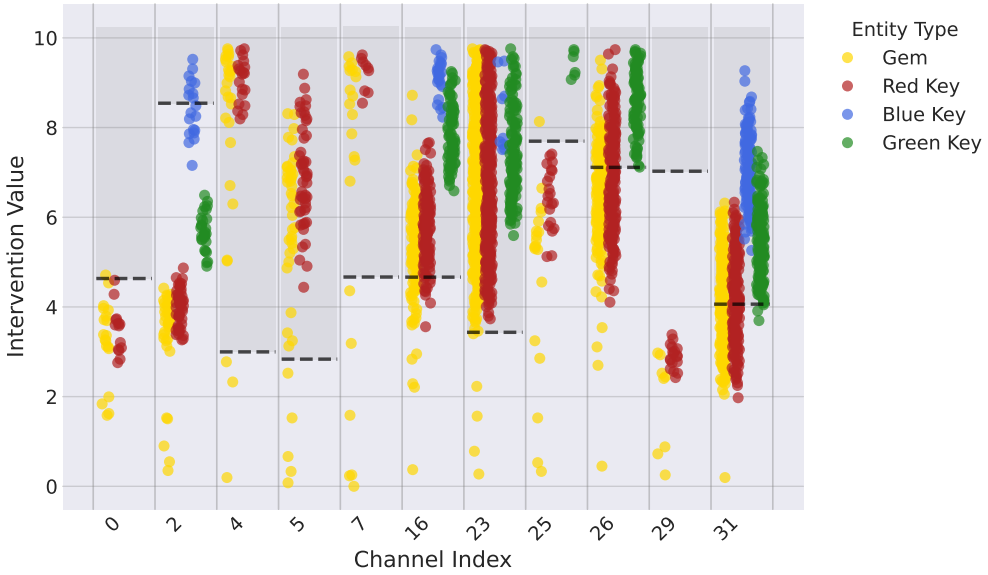

Figure 8: Steering intervention success rates for conv4a layer at checkpoint 45k. Colored dots represent successful steering interventions for different entities (blue key, green key, red key, gem), while black lines indicate the 99.9th percentile of normal activations. Each panel shows a different channel exhibiting steering behavior.

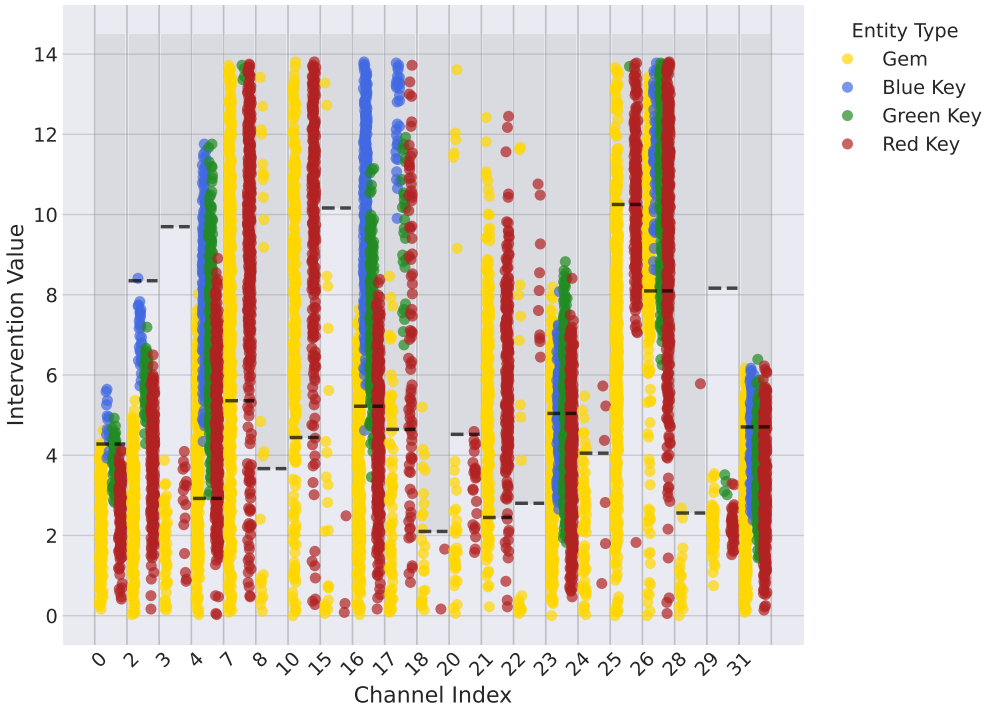

Figure 9: Steering intervention success rates for conv4a layer at checkpoint 50k. Colored dots represent successful steering interventions for different entities (blue key, green key, red key, gem), while black lines indicate the 99.9th percentile of normal activations. Each panel shows a different channel exhibiting steering behavior.

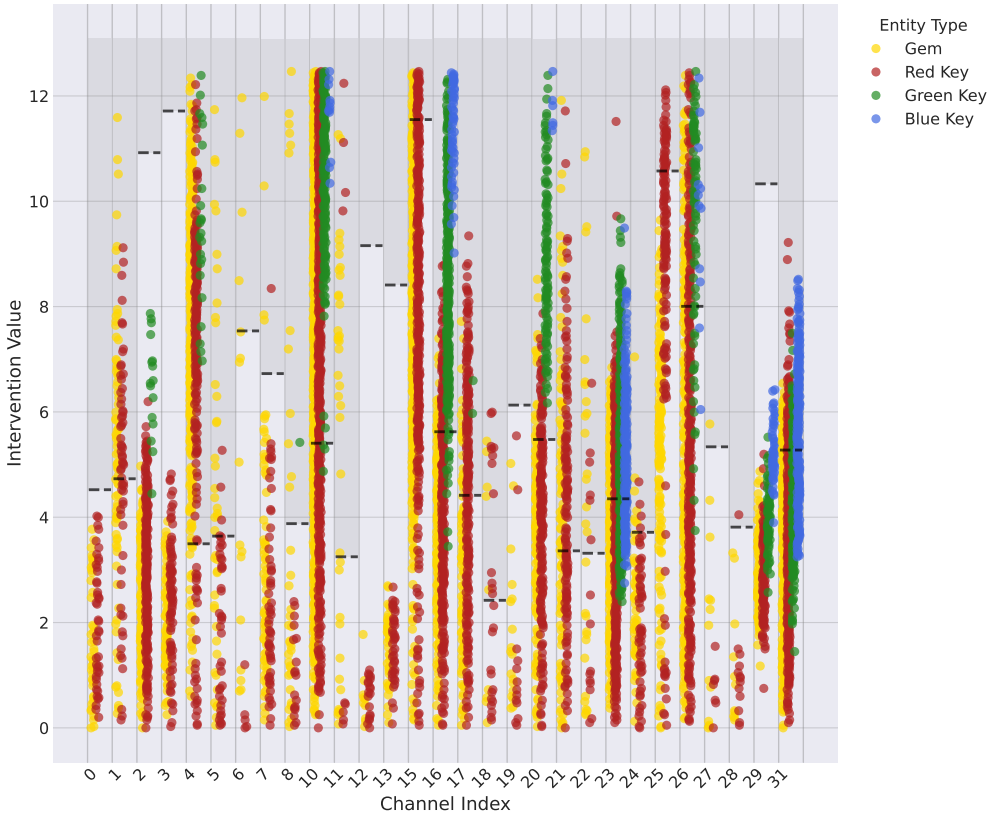

Figure 10: Steering intervention success rates for conv4a layer at checkpoint 55k. Colored dots represent successful steering interventions for different entities (blue key, green key, red key, gem), while black lines indicate the 99.9th percentile of normal activations. Each panel shows a different channel exhibiting steering behavior.

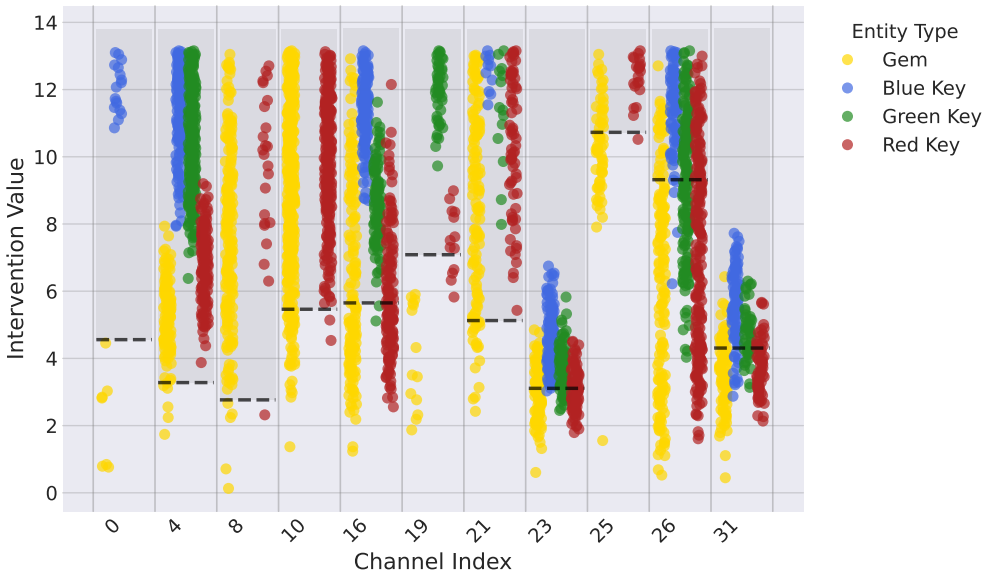

Figure 11: Steering intervention success rates for conv4a layer at checkpoint 60k. Colored dots represent successful steering interventions for different entities (blue key, green key, red key, gem), while black lines indicate the 99.9th percentile of normal activations. Each panel shows a different channel exhibiting steering behavior.

D Cross-Checkpoint Infrastructure Channel Analysis

Table 2 shows channel 25's impact across all tested checkpoints. Unlike other channels whose importance varies with representational drift, channel 25 has the greatest impact on model entity pickup rates when removed across checkpoints in post-convergence training.

D.1 Top 3 Most Critical Channels Per Checkpoint

Table 2: Top 3 most critical channels per checkpoint, ranked by average impact across all entities. Values show performance drop when channel is ablated. Channel 25 ranks #1 at all checkpoints.

Checkpoint	Rank	Ch	Gem	Blue	Green	Red	Avg
35k	1	25	18.0%	16.0%	16.0%	20.0%	**17.5%**
	2	29	2.0%	0.0%	6.0%	8.0%	4.0%
	3	7	0.0%	0.0%	-2.0%	16.0%	3.5%
40k	1	25	4.0%	8.0%	6.0%	16.0%	**8.5%**
	2	29	-2.0%	6.0%	12.0%	14.0%	7.5%
	3	1	-2.0%	6.0%	12.0%	4.0%	5.0%
45k	1	25	32.0%	12.0%	22.0%	8.0%	**18.5%**
	2	31	2.0%	-6.0%	8.0%	0.0%	1.0%
	3	28	0.0%	2.0%	0.0%	0.0%	0.5%
50k	1	25	**74.0%**	24.0%	30.0%	6.0%	**33.5%**
	2	26	0.0%	-4.0%	10.0%	10.0%	4.0%
	3	0	0.0%	0.0%	8.0%	0.0%	2.0%
55k	1	25	50.0%	18.0%	42.0%	18.0%	**32.0%**
	2	15	24.0%	8.0%	12.0%	12.0%	14.0%
	3	9	0.0%	4.0%	18.0%	6.0%	7.0%
60k	1	25	46.0%	20.0%	12.0%	8.0%	**21.5%**
	2	30	8.0%	8.0%	0.0%	12.0%	7.0%
	3	28	-2.0%	4.0%	16.0%	-2.0%	4.0%

E Universal vs Entity-Specific Infrastructure Channels

We apply our infrastructure experiments across different checkpoints post convergence to checkpoints at update steps 40k, 45k, 50k, 55k, and 60k, in addition to the 35k results presented above. While we observe shifts in which channels successfully steer the agent across checkpoints, there are some channels that seem to persist in their role across checkpoints.

This is particularly illustrated by channel 25, which ranks #1 in criticality at all 6 checkpoints tested (35k-60k). When ablated across checkpoints, it causes performance drops of 4-74% for gem, 8-24% for blue keys, 6-42% for green keys, and 6-20% for red keys. Averaged across all entities,its impact ranges from 8.5% (checkpoint 40k) to 33.5% (checkpoint 50k), peaking at checkpoint 50k with 74% reductions in gem pickup. This high performance across layers suggests that the channel does play a critical role, which is important enough to not be worth disrupting while changes occur elsewhere during training.

The channels that tend to have a high impact when removed mostly do not overlap with the independent navigation channels, with only one of the 32 channels, channel 9, appearing in the top 10 of either category. This indicates that the channels that are most critical for navigation rely on combined signals with other channels for success, while the independent channels may play a role of redundant navigation.

E.1 Infrastructure Channel Ablation Results

The results shown in Table 3 demonstrate the results of ablating the top 10 infrastructure channels for each entity type in the empty maze environment. We observe that ablating the channels identified as critical for gem collection had a far greater impact on the collection rates of other entities than removing their respective critical top 10 infrastructure channels. This may be the result of a greater share of the core navigational functionality being encoded in the gem infrastructure channels than in the key infrastructure channels. Interestingly, collection of the blue key was more greatly affected by the ablation of the red key's channels than by the ablation of its own infrastructure channels.

Table 3: Effect of ablating top 10 infrastructure channels for each entity on collection rates (n=400 trials per condition). Bold values indicate the largest performance decreases for each entity type. Values shown with 95% confidence intervals.

Condition	Gem	Blue Key	Green Key	Red Key
Baseline	74.5% $\pm$ 4.3%	98.5% $\pm$ 1.2%	95.0% $\pm$ 2.1%	80.0% $\pm$ 3.9%
Single Channel (Gem)	**27.3% $\pm$ 4.4%**	70.2% $\pm$ 4.5%	34.5% $\pm$ 4.7%	26.5% $\pm$ 4.3%
Single Channel (Blue Key)	32.5% $\pm$ 4.6%	**80.2% $\pm$ 3.9%**	52.8% $\pm$ 4.9%	38.0% $\pm$ 4.8%
Single Channel (Green Key)	23.2% $\pm$ 4.1%	97.5% $\pm$ 1.5%	**90.5% $\pm$ 2.9%**	51.7% $\pm$ 4.9%
Single Channel (Red Key)	49.8% $\pm$ 4.9%	55.2% $\pm$ 4.9%	49.2% $\pm$ 4.9%	**46.0% $\pm$ 4.9%**

Note: The diagonal shows on-target effects: Gem (-47.2%), Blue Key (-18.3%), Green Key (-4.5%), Red Key (-34.0%). Gem infrastructure channels affect all entities catastrophically, while key channels often have surprising and counter-intuitive impacts reducing their respective entities collection rates less than other entities.

F SAE Intervention Results

The following figures show steering intervention success rates for SAE latents both trained on the 35000 step checkpoint. Unlike the base model channels (which have 32 dimensions per layer), the SAE uses a 4x expansion factor, resulting in 128 latents per layer. These figures demonstrate that sparse autoencoder latents can also be successfully steered to influence agent behavior toward specific entities. Colored dots represent successful steering interventions for different entities (blue key, green key, red key, gem), while black lines indicate the 99.9th percentile of normal activations during standard gameplay. Figures shown have a threshold for inclusion of $\geq$ 200 successes for readability.

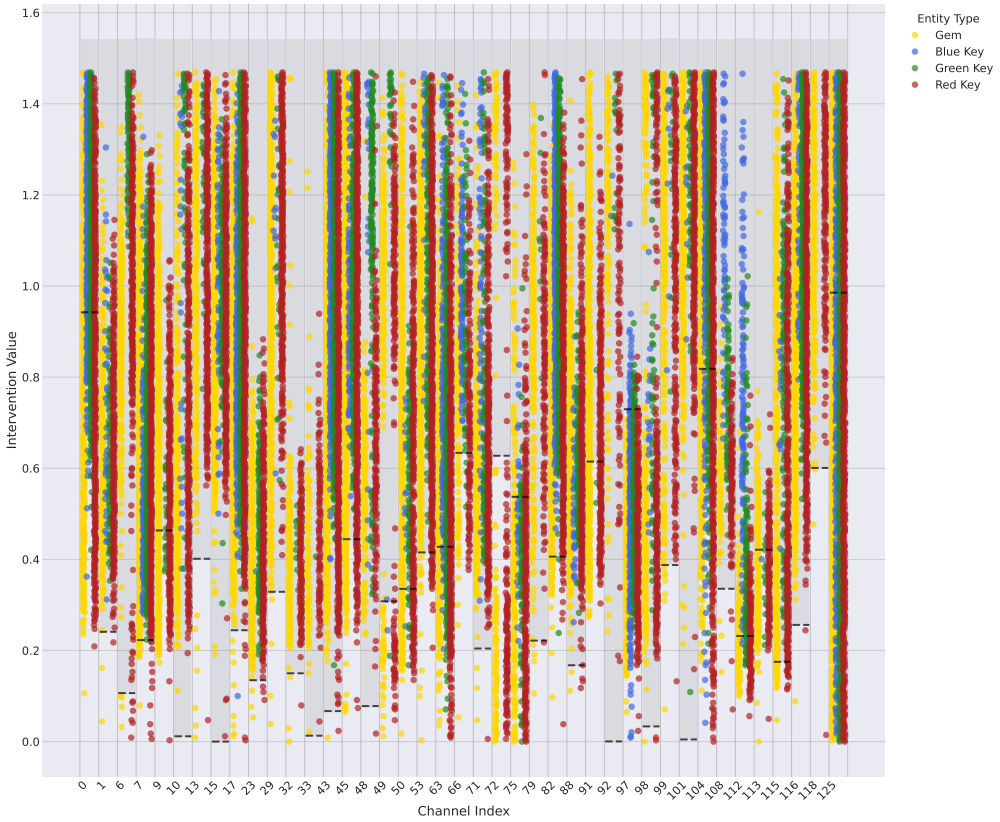

Figure 12: SAE latent steering intervention success rates for conv3a layer, showing the top 41 most successful latents with a cutoff threshold of ≥ 200 successes. Of particular note is the large number of steering successes in general in conv3a and also the significant overlap between different sections, including within similar activation regions, displaying relatively less of the multiplexing phenomena relative to elsewhere.

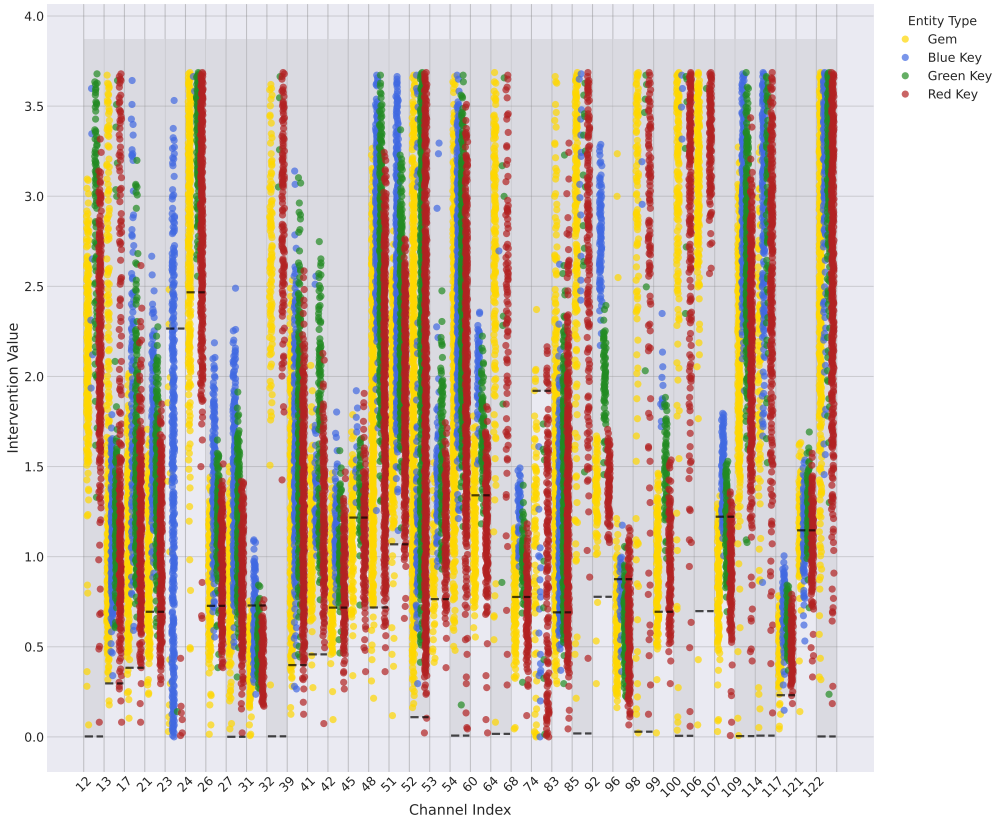

Figure 13: SAE latent steering intervention success rates for conv4a layer, showing the top 37 most successful latents with a cutoff threshold ≥ 200 successes. Conv4a SAE shows significantly sharper multiplexing than any other example, in particular latent 92 shows almost perfect delineation of entities. Latent 23 also contains an unusual capacity for steering the blue key in particular, indicating that some blue key specialisation may have been learned, though we caution against making strong conclusions from the steering alone due to the complex nature of the network.