Bilinear Convolution Decomposition for Causal RL Interpretability

Sinem Erisken
Independent

Alice Rigg
Independent

Narmeen Oozer
Martian

Abstract

Efforts to interpret reinforcement learning (RL) models tend to target the activation space, and fewer recent studies target the weight space. Here we use a dual framework of both the weight and activation spaces in order to interpret and intervene in a RL network. To enhance RL interpretability, we enable linear decomposition via linearization of an IMPALA network : we replace nonlinear activation functions in both convolution and fully connected layers with bilinear variants (we term BIMPALA). Previous work on MLPs have shown that bilinearity enables quantifying functional importance through weight-based eigendecomposition to identify interpretable low rank structure [Pearce et al., 2024b]. By extending existing MLP decomposition techniques to convolution layers, we are able to analyze channel and spatial dimensions separately through singular value decomposition. We find BIMPALA networks to be feasible and competitive, as they perform comparably to their ReLU counterparts when we train them on various ProcGen games. Importantly, we find the bilinear approach in combination with activation-based probing provide advantages for interpretability and agent control. In a maze-solving agent, we find a set of orthonomal eigenvectors (we term *eigenfilters*), the top-2 of which act as cheese (solution target) detectors, and another pair of eigenfilters we can manipulate to control the policy.

1 Introduction

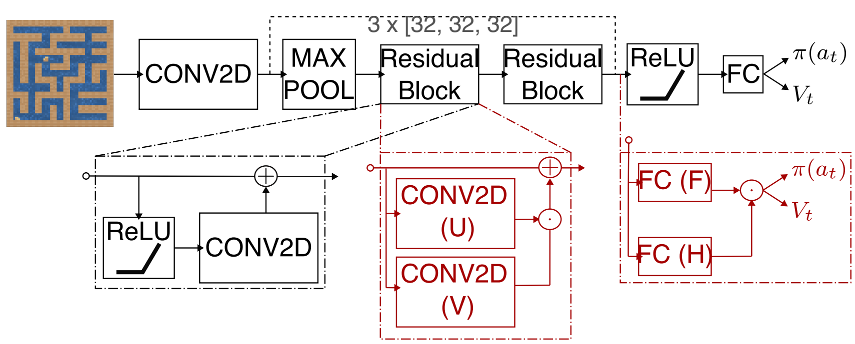

Figure 1: BIMPALA: a simplified IMPALA architecture (black) modified by replacing ReLU operations with bilinear gating (red) for both the convolution (CONV2D; Equation 3) and fully connected (FC; Equation 1) layers.

While recent advances in reinforcement learning have produced increasingly capable reasoning agents [Mnih et al., 2013, Gu et al., 2017, Baker et al., 2019], analyzing their internal mechanisms has proven difficult. This challenge is particularly pronounced in multi-step reasoning tasks, where the relationship between model architecture and computational strategy is often opaque. Additionally,

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

there is a general notion of the performance-interpretability trade-off [Assis et al., 2025], which argues that more transparent models tend to have lower performance.

We hypothesize that increased interpretability need not come at the cost of performance. We explore an approach embedded within mechanistic interpretability. Mechanistic interpretability has emerged as a promising framework for understanding neural networks by identifying and analyzing features-specific directions in activation space that encode meaningful computational patterns [Cunningham et al., 2023, Trenton Bricken et al., 2023, Adly Templeton et al., 2024, Rimsky et al., 2024]. Traditional approaches have focused primarily on activation patterns during inference, but recent work suggests that analyzing model weights directly may provide complementary insights.

Our work explores a subset of models where nonlinearities are replaced with linear counterparts. Bilinear MLPs [Dauphin et al., 2017] offer an architectural innovation that enables direct interpretation of model weights. While initially proposed for language modeling tasks [Pearce et al., 2024b], we show their benefits extend to understanding an agent's spatial decision-making. As proof of concept that the bilinear approach can indeed benefit interpretability of RL models, we simplified a common RL agent, IMPALA [Espeholt et al., 2018], and compared it with its bilinear counterpart (Figure 1).

We argue the importance of studying weights and activations jointly. Attribution approaches provide context-dependent heuristics, based on the data, to estimate causal relationships. Weight based analysis is a context-independent complement, where decomposition can allow us to validate attribution through the importance of components. By analyzing both the weight space through eigenfilters and the activation space through targeted probes, we find interpretable features that track specific computational steps, from interpretable convolution features to the action features. Additionally, we find that while standard basis analyses can appear informative, they often mask the true computational structure of the network. Instead, we show that bases informed by action spaces and targeted probes provide more reliable insights into model behavior.

Our contributions (1) We introduce a bilinear architectures (BIMPALA) for RL and show that a simplified model trains well in "easy" Procgen environments. (2) We show how bilinear convolution layers can be decomposed into basis of self interacting eigenfilters. (3) We show that the standard basis is often non-interpretable and less informative compared to basis derived from probes or the action/logits space. (4) We propose new techniques using weights alongside activations to analyze mechanisms in bilinear convolution networks. We validate our approach by finding a cheese detector on a maze solving agent and re-targeting the agent towards counterfactual cheese positions.

2 Background

2.1 Model and Environment

We chose IMPALA as our base model as it is a widely known RL architecture and was designed for both resource efficiency and scalability. The low rank structure of MLPs is already known to be interpretable, and is likely to also be the case for LSTMs which do not introduce new operators like a convolution. Our main objective in this paper was to evaluate the feasibility of interpretable RL with bilinear CNNs (bCNNs). We simplified the original large architecture IMPALA by (1) removing pooling layers and (2) omitting the LSTM after the FC layer , although note that LSTMs [Kim et al., 2018] can also be modified with bilinear gating (bLSTMs).

We believe our simplified BIMPALA architecture should, already, be adequate to perform competitively on Procgen games. Hybrid bCNN-bLSTMs have shown promise in medical imaging [Liu et al., 2024], and bCNNs in general image [Lin et al., 2017], classification tasks.

We chose games from Procgen [Cobbe et al., 2020], a suite of procedurally generated environments designed to benchmark efficiency and generalization of RL agents. As this work is proof of concept, we only trained our models on a random subset of games and we only considered "easy" mode.

2.2 Bilinear Gating

The core benefit of the bilinear approach hinges on removing nonlinearities from the neural network, allowing spectral decomposition. Spectral decomposition of MLPs has revealed interpretable low-rank structure across multiple tasks Pearce et al. [2024a], and we extend this approach to convolution layers for an RL agent.

In this section, we briefly review multi-layer perceptrons (MLPs), convolutions, and bilinear gating. Throughout, we denote scalars like s, vectors like $\mathbf{v}$, matrices like M, tensors like $\mathbf{T}$, dot product with $\cdot$, pointwise product with $\odot$, and convolution with $*$.

Bilinear MLPs A conventional MLP is composed of 3 (or more) fully-connected (FC) layers, where inputs are up-projected into a hidden layer and then down-projected into the output layer. The hidden activations of a conventional MLP can be characterized as a $\mathbb{R}^n \to \mathbb{R}^m$ encoder which takes input $\mathbf{x}$ and applies a learned linear transformation, with weights W and bias $\mathbf{b}$, followed by an activation function σ.

Modern models, such as LLMs, feature an encoder variant called a Gated Linear Unit (GLU), comprised of the pointwise product of *two* linear up-projections, with learned weights F and G, and only one of the projections is passed through an activation function. Omitting biases for simplicity,

$$\text{Enc}_{\text{GLU}}(\mathbf{x}, F, H) = \sigma(\mathbf{x}F) \odot (\mathbf{x}H)$$

Bilinear encoders, and our bilinear FC in Figure 1, use an identity activation, keeping the overall transformation linear:

$$\text{FC}_{\text{Bilinear}}(\mathbf{x}, F, H) = (\mathbf{x}F) \odot (\mathbf{x}H) \tag{1}$$

This linearization allows spectral decomposition of the weights and activations, which can have interpretable value [Pearce et al., 2024b]. Importantly, Pearce et al. [2024b] show bilinear MLPs can be expressed as a third order tensor $\mathbf{B}$, comprised of interaction matrices for each output dimension, parameterizing the interactions between pairs of inputs. In Decomposing Convolutions, we provide an analog $\mathbf{B}$ for convolution layers.

Convolution layers A 2D convolution layer (Conv2D) takes an input $\mathbf{X}$ of shape $[\text{width}, \text{height}, c_{in}]$ where c_{in} is the number of input channels, and applies a learned kernel $\mathbf{U}$ of width k with stride s followed by a pointwise activation function σ:

$$\text{Conv2D}(\mathbf{X}, \mathbf{U}) = \sigma(\mathbf{X} * \mathbf{U})$$

With $s = 1$, Conv2D outputs a tensor of shape $[\text{width}, \text{height}, , c_{out}]$, where c_{out} is the number of output channels. Kernel $\mathbf{U}$ of shape $[k, k, c_{in}, c_{out}]$ acts locally on $k \times k$ patches, and we denote the kernel for a given output channel i as $\mathbf{U^{(i)}}$. Assuming an identity activation and letting $\ell = \lfloor \frac{K}{2} \rfloor$, kernel weights $\mathbf{U^{(i)}}$, as illustrated in Appendix A Figure 12 (top left in blue), act on a local patch around spatial coordinates (α, β) via:

$$u(\alpha, \beta, i) = \sum_{j=1}^{c_{in}} \sum_{|k_1| \le \ell} \sum_{|k_2| \le \ell} U^{(i)}[j, k_1, k_2] \cdot X[j, \alpha + k_1, \beta + k_2] \tag{2}$$

Here, $u(\alpha, \beta, i)$ is a scalar, denoting output channel i's entry at spatial location (α, β), while $U^{(i)}[j, k_1, k_2]$ and $X[j, \alpha + k_1, \beta + k_2]$ are row and column vectors from $k \times k$ matrices representing the kernel and current input patch respectively for a single input-output channel combination.

Analogous to a bilinear FC (Equation 1), a bilinear convolution layer (BConv2D) would then require *two* convolutions. Assuming kernels $\mathbf{U}$ and $\mathbf{V}$,

$$\text{BConv2D}(\mathbf{x}, \mathbf{U}, \mathbf{V}) = (\mathbf{x} * \mathbf{U}) \odot (\mathbf{x} * \mathbf{V}) \tag{3}$$

3 Methods

3.1 Decomposing Convolutions

The main advantage of adopting the bilinear form for a convolution layer is decomposition into *sets* of orthonormal eigenvectors for each output channel, which we call *eigenfilters*. We can express a BConv2D layer as a tensor $\mathbf{B}$, comprised of interaction matrices (B) for each (scalar) output.

Specifically, B parameterizing the input channel interactions between pairs of inputs at a single spatial location for a single output channel (α, β, i) (Figure 12 in Appendix A).

Importantly, spectral decomposition is easily achievable because $\mathbf{B}$ has a symmetric form $\mathbf{B}^{\text{sym}}$. In Appendix A, we decompose $\mathbf{B}^{\text{sym}}$ for convolution layers and show it is equivalent to $\mathbf{B}$. In short, for each of c_{out} output channels, we get a matrix B^{sym} of dimension $k^2c_{in} \times k^2c_{in}$. Hence, each spatial location of the input image contributes to $\mathbf{B}$ with shape $[c_{out}, k^2c_{in}, k^2c_{in}]$. The gated operation, $\mathbf{B}^{\text{sym}}$, can be decomposed into scalar entries, b, for each output channel at spatial position (α, β):

$$b(\alpha, \beta, i) = \sum_{j=1}^{c_{in}} \sum_{z=1}^{c_{in}} \mathbf{x_j}^\top \mathbf{u_j}^\top \mathbf{v_z} \mathbf{x_z} \tag{4}$$

$\mathbf{u_j}$ is a flattened vector of the ith input channel of kernel $\mathbf{U}$ applied to the flattened input $\mathbf{x_j}$ at position (α, β). $\mathbf{v_z}$ is a similar vector from $\mathbf{V}$'s zth input channel applied to $\mathbf{x_z}$.

Bilinear decomposition into eigenfilters With the spectral theorem, matrix Q decomposes as $Q = F^\top \Lambda F$, where F is an orthonormal matrix (satisfying $F^{-1} = F^\top$) of eigenvectors, and Λ is a real, diagonal matrix of eigenvalues. Since convolution layers are connected locally, within and not across output channels, we choose our output directions in output channel space. We construct a tensor $\mathbf{Q}$ of interactions matrices, $\{Q^o\}_{o=1}^{c_{out}}$, by multiplying $\mathbf{B}^{\text{sym}}$ along a desired output direction $\mathbf{o} \in \mathbb{R}^{c_{out}}$.

For each output channel, $Q^o = \mathbf{o}\mathbf{B}^{\text{sym}}$. Q^o, shaped $[k^2c_{in}, k^2c_{in}]$, decomposes into an eigenbasis of k^2c_{in} filters, each shaped $[k, k, c_{in}]$, which we term *eigenfilters*. For the standard basis, $\mathbf{o} \in \{\mathbf{e}_i\}_{i=1}^{c_{out}}$. Note that we can consider the per-direction computation of Q^o matrices and their *eigenfilters* more generally: in an arbitrary basis, by multiplying B^{sym} with a transformation matrix R.

Contributions of eigenfilters To compute the contribution of an eigenfilter to an activation A, we apply the eigenfilter as a regular convolution filter to A. Since $B^{\text{sym}}(\mathbf{o}, \cdot, \cdot)$ is bilinear and symmetric, $Q^o(\mathbf{x}) := B^{\text{sym}}(\mathbf{o}, \mathbf{x}, \mathbf{x})$ is quadratic. So, the contributions of Q^o towards $\mathbf{o}$ for a flattened patch $\mathbf{x_{patch}}$ centered around a given position in A is: $Q^o(\mathbf{x_{patch}}) := \mathbf{x_{patch}}^\top F^\top \Lambda F \mathbf{x_{patch}} = (F\mathbf{x_{patch}})^\top \Lambda (F\mathbf{x_{patch}}) = \sum_{i=1}^{k^2c_{in}} \lambda_\mathbf{i} (f_i \mathbf{x_{patch}})^2$, where f_i is an individual eigenfilter with shape $[k, k, c_{in}]$.

As the eigenfilter activations are applied to every valid position uniformly, we can equivalently write $Q^o(A) = \sum_i^{k^2n} \lambda_\mathbf{i} (f_i * A)^2$.

3.2 Separating channels from spatial coordinates with SVD

The decomposition (subsection 3.1) gives us a full basis of eigenfilters along a direction in output channel space. If we are interested in the contribution of an activation or weight tensor $\mathbf{A}$, we can separate the output channels from the spatial dimensions, and we can use singular value decomposition (SVD) to determine the directions to decompose $\mathbf{B}^{\text{sym}}$ along the output channel space.

Consider a $[w, h, c_{out}]$ -shaped tensor , e.g. of activations, $\mathbf{A}$, reshaped into $[c_{out}, wh]$ as matrix A. Letting $d = wh$, we can decompose A via SVD:

$$A = S \Sigma V^\top = \sum_{i=1}^{d} \sigma_\mathbf{i} \mathbf{s_i} \mathbf{v_i}^\top \tag{5}$$

where S has shape $[c_{out}, c_{out}]$, and V has shape $[d, d]$. We can actually use SVD to decompose along the output directions of eigenvectors of any suitable A, including a probe. The (left) singular vectors $\mathbf{s_i}$ live in the output channel space, and can be used as output vectors along $\mathbf{B}^{\text{sym}}$. Since each singular vector $\mathbf{s_i}$ also has a singular value, s_i, we can aggregate the contributions of the singular values and the eigenfilters together when constructing the interaction matrix Q^A from interaction matrices $\{Q^s\}_{s=1}^{d}$ along the the singular vectors in output channel space:

$$Q^A(A) = \sum_{o=1}^{c_{out}} s_o Q^o(A) = \sum_{o=1}^{c_{out}} s_o \sum_{i=1}^{d} \lambda_i (f_i * A)^2 = \sum_{o=1}^{c_{out}} \sum_{i=1}^{d} s_o \lambda_i (f_i * A)^2 \tag{6}$$

Signed eigenvalues $s_o \lambda_i$ parameterize the importance, $|s_o \lambda_i|$, of an eigenfilter for its singular channel.

3.3 Procgen training

We trained our simplified IMPALA and BIMPALA models with proximal policy optimization (PPO), which tends to be effective and easy to tune [Schulman et al., 2017], for a subset of Procgen [Cobbe et al., 2020] environments: **Maze, Heist, Plunder**, and **DodgeBall**. Games were procedurally generated with "easy" distributions, which are computationally inexpensive and converge in less time steps than harder distributions. For full training parameters, see Table 1 in Appendix B.

3.4 Probe protocol

We suggest a protocol to enhance interpretability for RL with probes by connecting bottom-up mechanistic and top-down concept-based approaches.

1. Train a linear probe for a concept of interest on a Conv2D activation space with shape $[\mathrm{width}, \mathrm{height}, c_{out}]$, reshaped as $[c_{out}, \mathrm{width} \cdot \mathrm{height}]$
2. Decompose the probes (Equation 5), and use the top left channel-space singular vectors as output directions for the BConv2D layer of interest. Determine the number of singular components needed, based on the distribution of singular values.
3. Perform an eigendecomposition of the BConv2D layer (Equation 6) towards the top left singular vectors in channel space, to identify directions in the filter weights that write to the probe (similar to Pearce et al. [2024b]).

This protocol will yield a full basis of eigenvectors for each output direction.

Cheese probe In subsection 4.2, We design a cheese probe as a case study of our probe protocol (subsection 3.4). We trained linear probes to detect the presence of the cheese at position $(8, 14)$ in the maze by creating a dataset comprising 2000 pairs of mazes, one with with the cheese at position $(8, 14)$ and the other without a cheese.

We trained position probes on a BConv2D activation space with shape $[w, h, c_{out}]$, reshaped as $[w, h, c_{out}]$ on the output of the BConv2D layer of interest. We decomposed the probe's weights via SVD (Equation 5). We could then use the top (left) singular vectors as output directions along the BConv2D layer of interest. For probe results, see Appendix C.

3.5 Ablations

We perform 3 separate sets of ablation experiments: (1) for the cheese probe, (2) in the standard basis and (3) for action features. In each set of experiments, we run multiple ablations. For the probe and standard basis experiments, we run, separately for each k, ablations for all but the top-k eigenfilters for each output channel of selected BConv2D layers. For ablation within the standard basis and action features, we also ablate everything but the top-k eigenfilters in the FC layer. For each ablated model, we reconstruct the network using the top-k eigenfilters and then run the model in the Maze environment. To ensure we are comparing the ablated models fairly within each set of ablation experiments, we run the same mazes, using 20 seeded environments, capping steps per rollout at 200 to save runtime.

3.6 Steering

While the FC layer outputs the value and policy, we are interested in the contribution of the convolution layers to solving the task. As such, we leave the FC layer intact and try to the steer the model from the convolution layers alone. We modified steering examples following Mini et al. [2023]. Rather than averaging activation spaces together, we directly alter the weight contributions from hidden layers (Res in Equation 7).

Our is aim is to re-target the agent from the maze's actual location towards a counterfactual cheese location x'. To do so, we first obtain the activations for the maze's cheese position (x_{cheese} in Equation 7) by subtracting the activations for the maze without the cheese from the activations for the maze with the cheese. Similarly, we get the activations for the counterfactual cheese position (x'_{cheese} in Equation 7) by subtracting the activations for the maze without the cheese from the activations for the maze with the cheese in the counterfactual position. We intervene using the top-2 eigenfilters (eig in Equation 7) and overwrite the contributions, using the equation:

$$Res' = Res - eig * (x_{\text{cheese}}) + eig * (x'_{\text{cheese}}) \tag{7}$$

4 Experiments

In order to evaluate the usefulness of the bilinear approach in the context of RL, we ran a series of experiments. We detail training procedures, experimental protocols, and key findings from both quantitative and qualitative perspectives. We ask (1) do bilinear architectures achieve competitive performance compared to standard models like ReLU-based IMPALA and (2) do bilinear layers provide interpretable representations through spectral decomposition and probe-based analyses?

In order establish feasibility, we first evaluate and compare performance between BIMPALA and IMPALA on a handful or randomly selected "easy" ProcGen environments.

We next train probes and propose a protocol to decompose probes in conjunction with convolution layers. This allows us to identify a cheese filter using the top-2 eigenfilters of a convolution layer.

We next explore methods without the need for training probes. First, we turn to the standard channel basis and perform ablation experiments. Unfortunately, we do not find the standard basis alone to be informative enough for interpretability.

We then decide to adopt two different approaches using both weights and activations without training probes. First, we decompose the full connected layer along each policy action and perform ablation experiments. We find action features to be faithful to actions needed to solve the maze. Finally, we perform steering experiments where we re-target the agent towards a counterfactual cheese position.

4.1 BIMPALA matches IMPALA performance

Architecture baseline We adapted the existing IMPALA framework Espeholt et al. [2018] by (1) simplifying the network by removing some convolution layers so that the residual block is a simple gated convolution with a skip connection as well as removing the LSTM layer after the FC layer and (2) modifying the original structure to incorporate bilinear gating mechanisms in both Conv2D and FC layers (Figure 1). We refer to the bilinear variant as BIMPALA (Bilinear IMPALA).

Evaluation As proof of concept for interpretable bilinear RL, we trained (Methods subsection 3.3) simplified IMPALA and BIMPALA alongside each other on the "easy" distributions of some (**Maze, Heist, Plunder**, and **DodgeBall**) of the procedurally generated environments within the established Procgen benchmark[Cobbe et al., 2020].

We find BIMPALA matched and occasionally outperformed IMPALA across the environments we tested (Figure 2), validating the feasibility of using bCNNs for RL tasks. Specifically, BIMPALA generally demonstrated faster learning, higher final performance in terms of expected return, and maintaining lower entropy.

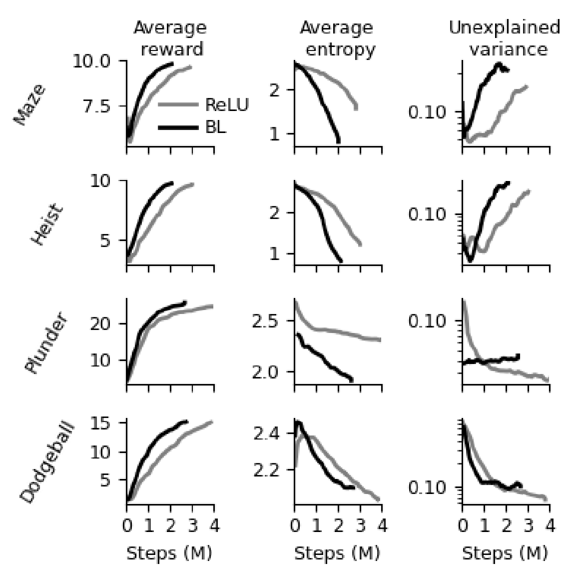

Figure 2: ReLU and Bilinear IMPALA perform comparably across different ProcGen environments.

4.2 Protocol to enhance interpretability for RL with probes

Having established that the bilinear approach can perform competitively in RL environments, we next want to use this architecture to enhance interpretability. In subsection 3.4, we describe a protocol to yield a full basis of eigenfilters for each output direction relevant for a concept of interest. Briefly, we train a linear probe for a concept of interest on a Conv2D activation space. With SVD (subsection 3.2), we can identify the probe's top singular vectors, which we can use as output directions for a BConv2D layer. We can then perform eigendecomposition towards the top left singular vectors in output channel space to identify eigenfilters which contribute to the probe.

It's possible for the important eigenvectors between output directions to not be fully orthogonal, especially if interpreting multiple probes in parallel. Although we do not investigate overlapping filters here, analyzing the cosine similarity between important (as measured by $|s_j \lambda^i_{u_j}|$) eigenvectors relating to different singular channels may further inform the function of the eigenvectors.

4.3 Case study: cheese probe

With the protocol defined, the next step is to implement it by training concept probes for specific features and analyzing their decomposition. For the remainder of the paper, we focus on Maze, where the player, a mouse, must navigate a maze to find the sole piece of cheese and earn a reward. We generate a dataset of sets of mazes with and without a cheese and train linear probes to detect the presence of the cheese at some position subsection 3.4.

Probes trained well on the outputs of the residual blocks, and had $> 99\%$ accuracies and F_1 scores (Table 2 in Appendix C).

Dominant singular probe channels Decomposing the probe, we see a spectra (Figure 3). The top singular component alone explains 30% of the variance, and 16 components are needed to explain $\geq 90\%$ of the variance.

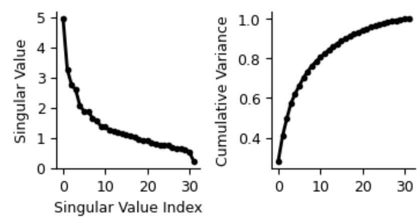

Figure 3: Singular values and explained variance for cheese probe

Eigenfilter decomposition for singular probe channels Decomposing the last BConv2D layer towards the top (left) singular channel, we see the standard channel basis spectra has just two eigenvalues (Figure 4, right). The singular spectrum for the cheese probe (Figure 4, left), however, was nondegenerate, and thus more likely to be informative.

In order to verify the importance of the singular spectrum for solving the maze, we repeated the decomposition for the first and second BConv2D layers and performed ablation (subsection 3.5). We ablated all but the topk eigenfilters for each output channel of each of the BConv2D layers. We reconstructed our networks and ran each of ablated models on the same 20 Maze environments. (Figure 5).

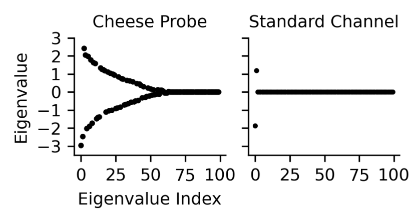

Figure 4: Spectral decomposition of the final BConv2D towards the cheese probe's top singular output channel (left) and in the standard channel basis (right)

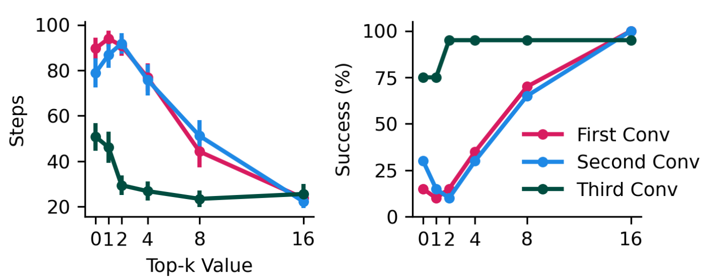

Figure 5: metrics when ablating parts (bottom - k+1) of different convolution layers' eigenfilter spectrum associated with the cheese probe. Here and subsequent plots, error bars are SEM

We find that maze performance recovers close to 100% with just the top-2 eigenfilters in the last BConv2D layer. In this last convolution layer, additional eigenfilters help solve the maze in less steps. For the first and second BConv2D layers, we see a different trend, where it takes 16 eigenfilters for maze performace to be recovered, and the contributions of each added eigenfilter is less step-like and more continuous. This continuous distribution may suggest that decomposing the layers towards the cheese probe's top singular channel is informative.

In fact, in the second BConv2D layer, we can already find information about the cheese location already from just the top positive eigenfilter. In Figure 6, we visualize the top positive and negative eigenfilter activations for a set of pairs of mazes, one with the cheese at the selected position and the other without the cheese. While the positive filter activates on non-cheese patterns, the negative filter down-weighs non-cheese patterns without erasing the cheese activation. The positive and negative activations of the respective filters result in a cheese detector filter. We found similar cheese filters when we considered other mazes and BConv2D layers.

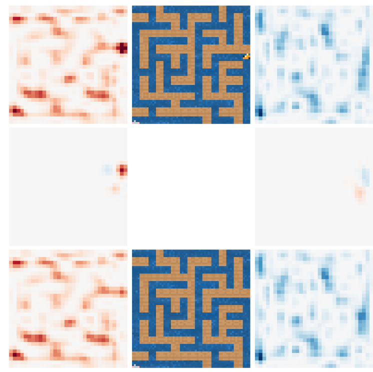

Figure 6: Activations for the top positive (left) and negative (right) eigenfilters in the second BConv2D layer, for the cheese probe's top singular channel. Activations for a maze with cheese (top) vs without cheese (bottom). Middle plots show the difference between the activations with and without cheese.

4.4 Ablation within the standard bases

While the probe approach is rather promising and we are successful in finding a cheese detector in the second convolution layer, it may not be feasible nor scalable to train probes for each feature we may want to interpret. This is especially true once we move beyond toy-like tasks such as the Maze environment. Hence, despite the degenerate spectrum (Figure 4), we turn back to the standard basis.

We ask how many eigenfilters are necessary for performance? We ablated all but the topk eigenfilters for the FC layer, all the BConv2D layers, or just the last BConv2D layer. As we may have predicted from Figure 4, the spectrum of the last BConv2D layer is not informative and is marginally necessary for full performance. The agent, when compared to the BIMPALA, has a similar success rate, in a relatively low number of extra steps, and receives similar rewards when we ablate the last BConv2D entirely (topk=0), (Figure 7).

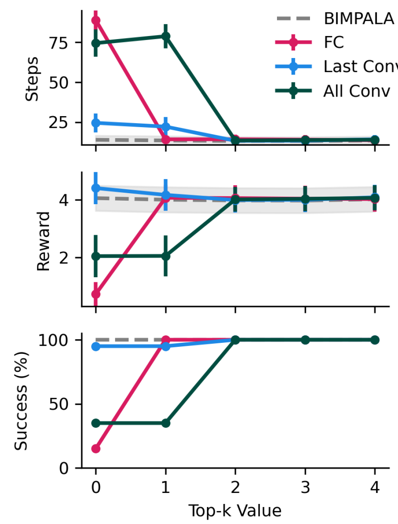

Figure 7: Maze metrics during standard basis ablation.

When we ablate all the BConv2D layers together, we see that the top-2 eigenfilters (per output channel) are sufficient to recover full performance (Figure 7). Full performance in the FC layer is achieved with the top eigenfilter. This eigenfilter is more important for performance than all the BConv2D layers combined, resulting in lower success in more steps with less reward when ablated.

While the contribution of the FC layer relative to the BConv2D layer may be an important insight, it is also expected. We could expect the FC , as the last layer of the network that outputs the policy and value, to contain most the information about the next step and therefore be the largest contributor to performance. Beyond that, the standard channel basis *by itself* may not be very fruitful for decomposing the network for interpretability. For example, while we can deduce that in the last BConv2D layer, the top positive and negative eigenfilters work together (Figure 4, 7), we do not know anything more granular.

4.5 Interpretability based on action features

For example, Figure 8 shows the eigenvalues in the action spectrum for the UP action. We can see many UP action eigenvectors in the FC layer, with one very large positive eigenvalue. We next used these action spectra of the FC layer in ablation experiments. Decomposition may still be useful for interpretability beyond the standard basis and without training probes. The FC layer outputs the directions of movement for the policy (UP, DOWN, RIGHT, LEFT). Instead of training probes,

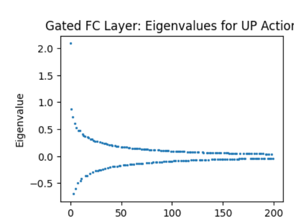

Figure 8: UP spectrum

we could alternatively decompose the directions relevant for actions directly by decomposing in the direction of each action output.

Ablation In ablation experiments, we found that, despite the dense spectrum (Figure 8), preserving the top eigenvector for each action in the FC layer was sufficient for a 100% success rate (Figure 13 in Appendix E). When we inspected impact of the eigenvectors by visualizing the vector fields, we saw the same trends across the multiple mazes and directions we inspected.

In Figures 9- 10, we show example vector fields for ablations in two action spectra, UP and LEFT. In Figure 9, we see how a single UP eigenvector proves sufficient to encode the optimal path through the maze. Specifically, we see that the upward logit values are selectively increased along the solution path and suppressed near dead ends. Similarly, in Figure 10 we see the effect of the LEFT action spectrum. Note that the mouse, who is typically located at the bottom left corner at the start of an episode, can solve the maze without any LEFT eigenvectors (Figure 10, Top 0 LEFT). Yet, we see that increasing the number of LEFT eigenvectors allows the agent to reach the cheese from other locations, such as at the top. As we add more left eigenvectors, we see that the misleading right arrows diminish and the agent gradually reconstructs its left arrows (Figure 10, Diff: Top 4 from ALL LEFT), making the maze solvable for more configurations.

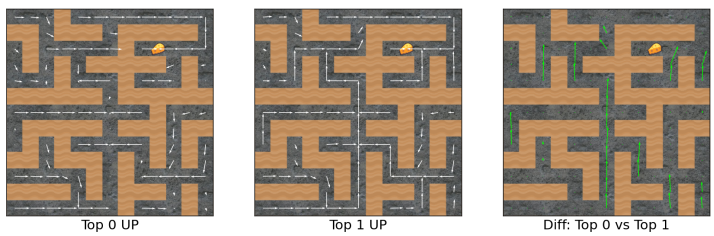

Figure 9: Vector field visualizing maze navigation without any (left) and with a single (middle) UP eigenvector, and the difference highlighted in green (right). The top UP eigenvector is sufficient to solve the maze and without UP eigenvectors, the agent does not move upwards

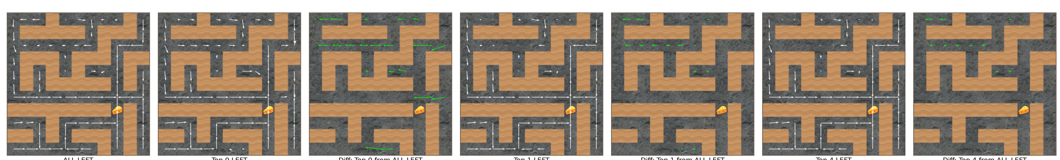

Figure 10: LEFT action spectrum visualization in a maze environment. While the maze is solvable without any LEFT eigenvector, adding LEFT eigenvectors allows solving the maze from other starting positions (e.g. top or right of the cheese)

4.6 Steering

Having seen the effect of action spectra on maze solving and the importance of the FC layer to solving the maze in general, we wondered if it was possible to redirect the agent while leaving the FC intact. In particular, we wondered if we could redirect the agent by intervening in the convolution layers.

Our steering experiment (subsection 3.6) aims to re-target the agent away from the actual cheese position (cheese in Figure 11) towards a counterfactual cheese position (red dot in Figure 11). We followed the steering examples of Mini et al. [2023], with a modification: rather than averaging activation spaces together, we directly alter the weight contributions from BConv2D layers.

We obtain the activations for the maze's cheese position (x_{cheese}) by subtracting the activations for the maze without the cheese from the activations for the maze with the cheese at its actual location. Similarly, we get the activations for the counterfactual cheese position (x'_{cheese}) by subtracting the activations for the maze without the cheese from the activations for the maze with the cheese in the counterfactual position. We intervene by overwriting the contributions on the BConv2D layer by subtracting the top-2 eigenfilters of x_{cheese} and adding the top-2 eigenfilters of x'_{cheese}.

We ran our steering experiment on various different counterfactual cheese positions, and had similar qualitative results, which we share here via an example (Figure 11), where the counterfactual cheese position is on the opposite side of the maze (red dot in Figure 11). In this example, the steering was successful and mouse could not solve the maze. We can see that the vector fields indicating movement are altered. Specifically, we can see arrows pointing towards the counterfactual cheese position during intervention (Figure 11, middle) lead the mouse from the bottom left towards the red dot. And if we look at the difference between the original and intervened mazes, we can see that the green arrows draw paths away from the real cheese towards the counterfactual cheese position.

5 Discussion

Summary We introduce an approach to interpreting convolution neural networks, by replacing nonlinearities with bilinear variants that achieve comparable and occasionally superior performance. Our approach allows us to find a closed form for self-interacting convolution features that can be combined with a top down concept based approach to derive causally relevant mechanisms used by RL agents in their decision making process. Through decomposition, we were able to find the relevant component and use them to identify the cheese target in a Maze environment. We were able to also show a causal relationship between the representation of the cheese in the top components of the convolution layers by steering the mouse towards a counterfactual cheese position. In short, we see great potential and value in bilinear variants that offer more interpretability prospects while achieving competitive performance to their non-analytic variants.

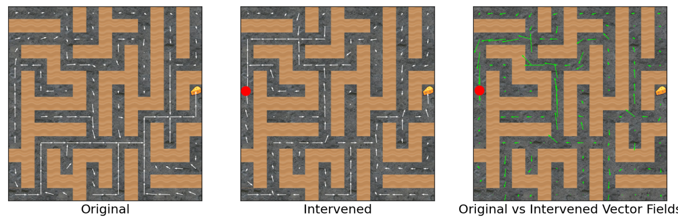

Figure 11: Re-targeting the agent by intervening to redirect towards a counterfactual cheese position (red dot)

Future work As proof of concept, the scope of this paper is rather limited and leaves room to expand the methods and generalize interpretability approaches. A significant addition to our interpretability methods would be to incorporate multiple steps in the RL task, enabling us to track eigenfilters across decision steps. This could easily be done even in the Maze environment by using Procgen's "memory" mode where the mouse gets a partial view of the maze while exploring.

While we do not study models featuring batch norms in this paper, we believe they can be readily incorporated into our framework. During inference, batch normalization applies a fixed affine transformation to activations, which is compatible with our decomposition approach. Additionally, our work currently uses max pooling, which is performant but not analytically decomposable and thus remains a blackbox in our work. Future work should explore alternative pooling operations that are both analytically tractable and performant, enabling end-to-end decomposition of the full architecture.

Limitations We found significant challenges in interpreting the units of computation in an entirely data independent fashion. Instead, we found that top activating dataset examples for eigenfilters tend not to be informative. Still, the decomposition allows us to break concept probes into more granular units of computation. We considered only one architecture, IMPALA, for our policy, although we expect the general approach of replacing nonlinearities with bilinear variants to be widely applicable. Due to computational requirements, we trained on the "easy" mode of a handful of ProcGen environments and we only analyzed the BIMPALA network for interpretability in the context of the Maze environment. It is not clear if the methods we presented here will transfer well to more complex environments with multiple objectives. Studying activations of probes, eigenvectors and eigenfilters across the temporal dimension may help in identifying interesting phenomenon such as reasoning and planning in RL environment. However, this might not be tractable with our current method as interactions between eigenfilters grow exponentially with each time step. Additionally, we do not concretely show how to derive insights specifically for multi-step processes, and aim to address this in future work. Similarly, we do not address a range of components often found in convolution neural networks, such as batch norm, dropout, or pooling. Their implications, such as the performance tradeoffs between different pooling strategies, should be considered when evaluating architecture variants in the future.

References

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus, Tristan Hume, Francesco Mosconi, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet, May 2024. URL `https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html`. tex.ids= zotero-348.

André Assis, Jamilson Dantas, and Ermeson Andrade. The performance-interpretability trade-off: A comparative study of machine learning models. *Journal of Reliable Intelligent Environments*, 11 (1):1, 2025.

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *International conference on learning representations*, 2019.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning, 2020. URL `https://arxiv.org/abs/1912.01588`.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse Autoencoders Find Highly Interpretable Features in Language Models, October 2023. URL `http://arxiv.org/abs/2309.08600`. arXiv:2309.08600 [cs].

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks, 2017. URL `https://arxiv.org/abs/1612.08083`.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018. URL `https://arxiv.org/abs/1802.01561`.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

Chanho Kim, Fuxin Li, and James M Rehg. Multi-object tracking with neural gating using bilinear lstm. In *Proceedings of the European conference on computer vision (ECCV)*, pages 200–215, 2018.

Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnns for fine-grained visual recognition, 2017. URL `https://arxiv.org/abs/1504.07889`.

Shan Liu, Jiang Wang, Shanshan Li, and Lihui Cai. Multi-dimensional hybrid bilinear cnn-lstm models for epileptic seizure detection and prediction using eeg signals. *Journal of Neural Engineering*, 21(6):066045, 2024.

Ulisse Mini, Peli Grietzer, Mrinank Sharma, Austin Meek, Monte MacDiarmid, and Alexander Matt Turner. Understanding and Controlling a Maze-Solving Policy Network. October 2023. URL `https://openreview.net/forum?id=vNkUeTUbSQ`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Michael T. Pearce, Thomas Dooms, and Alice Rigg. Weight-based Decomposition: A Case for Bilinear MLPs, June 2024a. URL `http://arxiv.org/abs/2406.03947`. arXiv:2406.03947 [cs].

Michael T. Pearce, Thomas Dooms, Alice Rigg, Jose M. Oramas, and Lee Sharkey. Bilinear MLPs enable weight-based mechanistic interpretability, October 2024b. URL `http://arxiv.org/abs/2410.08417`. arXiv:2410.08417 [cs].

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering Llama 2 via Contrastive Activation Addition, March 2024. URL http://arxiv.org/abs/2312.06681. arXiv:2312.06681 [cs].

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Trenton Bricken, Adly Templeton*, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Chris Olah. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning, October 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

A Tensor Decomposition of Bilinear Convolution Tensors

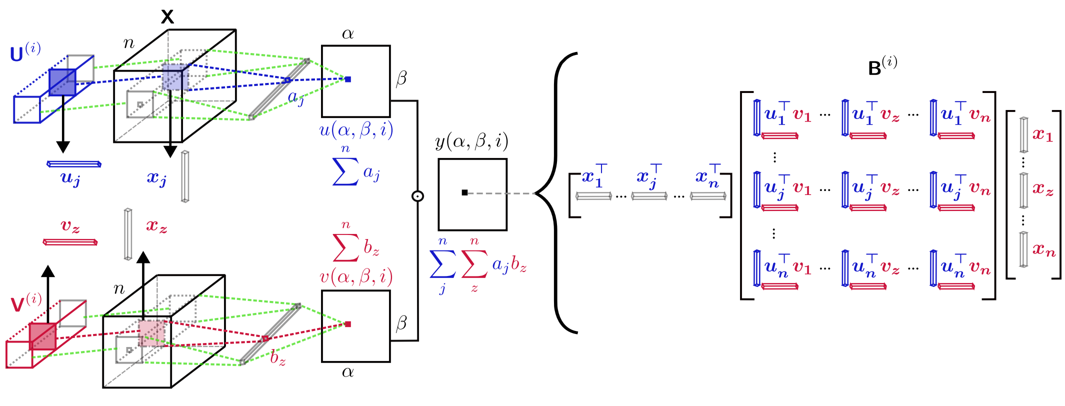

Figure 12: Transformation from spatial convolution operations (left) to a bilinear interaction matrix B (right) associated with scalar entry $y(\alpha, \beta, i)$ in output channel i. The diagram emphasizes how local spatial convolutions (shown in the cubes) are transformed into a bilinear form B. (Left, top, blue) Computation of spatial convolutions $\mathbf{U}^{(i)}$ with input $\mathbf{x_j}$, producing terms a_j. (Left, bottom, red) Computes convolutions $\mathbf{V}^{(i)}$ with input $\mathbf{x_k}$, producing terms b_k. (Right) The previous operations can be reformulated as a product of three block matrices, where the outer product of input channel responses $(\mathbf{u}^\top \mathbf{v})$ forms a bilinear matrix, $(\mathbf{B}^{(i)} = B)$. We show in this appendix that B has a symmetric form B^{sym}
.

As in Equation 2, consider the output of a convolution layer at location (α, β) for the i-th output channel:

$$u(\alpha, \beta, i) = \sum_{j=1}^{c_{in}} \sum_{|k_1| \le \ell} \sum_{|k_2| \le \ell} U^{(i)}[j, k_1, k_2] \cdot X[j, \alpha + k_1, \beta + k_2]$$

We define the contribution of the filter applied to the jth input channel as

$$a_j = \sum_{|k_1| \le \ell} \sum_{|k_2| \le \ell} U^{(i)}[j, k_1, k_2] \cdot X[j, \alpha + k_1, \beta + k_2] \quad ,$$

which contributes to the ith channel output of the first convolution as

$$u(\alpha, \beta, i) = \sum_{j=1}^{c_{in}} a_j$$

We consider a flattened vector $\mathbf{x_j}$: the input tensor $X[j, \alpha : \alpha + k, \beta : \beta + k]$ flattened into a k^2-dimensional vector for each spatial location (α, β). Similarly, $\mathbf{u_j}$ is the flattened filter $U^{(i)}[j, :, :]$. Note tha the filter is independent of the position (α, β).

Using these flattened representations, we can express a_j as

$$a_j = \mathbf{u_j} \cdot \mathbf{x_j}$$

Note that we have removed the notation for the output channel i, as all the operations we discuss here are for a single output channel.

As the gated operation is given by

$$y(\alpha, \beta, i) = u(\alpha, \beta, i) \odot v(\alpha, \beta, i),$$

we can,similarly, consider the second convolution's k^2 row vector $\mathbf{v_z}$ and k^2 column vector $\mathbf{x_z}$, which define the contribution to the zth input channel as:

$$b_z = \mathbf{v_z} \cdot \mathbf{x_z}$$

and ith channel output of the second convolution as :

$$v(\alpha,\beta,i)=\sum_{z=1}^{c_{in}} b_z$$

For simplicity, we will assign c_{in} to n. The gated operation is then :

$$y(\alpha,\beta,i)=\sum_{j}^{n}\sum_{z}^{n} a_j b_z$$

We note that each term, a_j and b_z, is a scalar and is therefore equal to its transpose. Now lets consider the **interaction** between input channels j and z:

$$\begin{aligned} a_j b_z &= (\mathbf{u_j x_j})(\mathbf{v_z x_z}) \quad \text{. Substituting } a_j \text{ and rearranging terms:} \\ &= (\mathbf{x_j}^\top \mathbf{u_j}^\top)(\mathbf{v_z x_z}) \end{aligned}$$

This yields the gated operation:

$$y(\alpha,\beta,i)=\sum_{j}^{n}\sum_{z}^{n} \mathbf{x_j}^\top \mathbf{u_j}^\top \mathbf{v_z x_z} \tag{8}$$

We can write the sum $\sum_j^n \sum_z^n \mathbf{x_j}^\top \mathbf{u_j}^\top \mathbf{v_z x_z}$ as a product of three block matrices (Figure 12):

$$\begin{bmatrix} \mathbf{x_1}^\top & \cdots & \mathbf{x_j}^\top & \cdots & \mathbf{x_n}^\top \end{bmatrix} \begin{bmatrix} \mathbf{u_1}^\top \mathbf{v_1} & \cdots & & \mathbf{u_1}^\top \mathbf{v_n} \\ \vdots & \ddots & & \vdots \\ & & \mathbf{u_j}^\top \mathbf{v_z} & \\ \vdots & & \ddots & \vdots \\ \mathbf{u_n}^\top \mathbf{v_1} & \cdots & & \mathbf{u_n}^\top \mathbf{v_n} \end{bmatrix} \begin{bmatrix} \mathbf{x_1} \\ \vdots \\ \mathbf{x_z} \\ \vdots \\ \mathbf{x_n} \end{bmatrix}$$

The middle matrix is the **interaction matrix** B, and it has a symmetric form given by :

$$B^{\text{sym}} = \begin{bmatrix} & \cdots & \cdots & \\ \vdots & \ddots & & \vdots \\ & \frac{\mathbf{u_j}^\top \mathbf{v_z} + (\mathbf{u_z}^\top \mathbf{v_j})^\top}{2} & & \\ \vdots & \cdots & \ddots & \vdots \end{bmatrix}$$

In Equation 8, each term, $\mathbf{x_j}^\top \mathbf{u_j}^\top \mathbf{v_z x_z}$, is a scalar and is therefore equal to its transpose. With this in mind, we show that B^{sym} is symmetric over all possible input pairs (j,z). That is,

$$\mathbf{x_j}^\top B^{\text{sym}}[j,z]\mathbf{x_z} = \mathbf{x_z}^\top B^{\text{sym}}[z,j]\mathbf{x_j} \quad ,$$

where

$$\mathbf{x_j}^\top B^{\text{sym}}[j,z]\mathbf{x_z} = \mathbf{x_j}^\top \left(\frac{\mathbf{u_j}^\top \mathbf{v_z} + (\mathbf{u_z}^\top \mathbf{v_j})^\top}{2} \right) \mathbf{x_z} \quad ,$$

and

$$\begin{aligned} \mathbf{x_z}^\top B^{\text{sym}}[z,j]\mathbf{x_j} &= \mathbf{x_z}^\top \left(\frac{\mathbf{u_z}^\top \mathbf{v_j} + (\mathbf{u_j}^\top \mathbf{v_z})^\top}{2} \right) \mathbf{x_j} \quad \text{. Transposing and rearranging terms} \\ &= \mathbf{x_j}^\top \left(\frac{\mathbf{v_j}^\top \mathbf{u_z} + \mathbf{u_j}^\top \mathbf{v_z}}{2} \right) \mathbf{x_z} \\ &= \mathbf{x_j}^\top \left(\frac{\mathbf{u_j}^\top \mathbf{v_z} + (\mathbf{u_z}^\top \mathbf{v_j})^\top}{2} \right) \mathbf{x_z} \\ &= \mathbf{x_j}^\top B^{\text{sym}}[j,z]\mathbf{x_z} \end{aligned}$$

Last, we show that B agrees with B^{sym} on every pair of inputs. In other words, we will show

$$\mathbf{x_z}^\top B^{\text{sym}}[z,j]\mathbf{x_j} + \mathbf{x_j}^\top B^{\text{sym}}[j,z]\mathbf{x_z} = \mathbf{x_z}^\top B[z,j]\mathbf{x_j} + \mathbf{x_j}^\top B[j,z]\mathbf{x_z}$$

Starting with the RHS

$$\begin{aligned}
\mathbf{x_z}^\top B^{\text{sym}}[z,j]\mathbf{x_j} + \mathbf{x_j}^\top B^{\text{sym}}[j,z]\mathbf{x_z} & \\
&= \mathbf{x_z}^\top \left(\frac{\mathbf{u_z}^\top \mathbf{v_j} + (\mathbf{u_j}^\top \mathbf{v_z})^\top}{2}\right)\mathbf{x_j} + \mathbf{x_j}^\top \left(\frac{\mathbf{u_j}^\top \mathbf{v_z} + (\mathbf{u_z}^\top \mathbf{v_j})^\top}{2}\right)\mathbf{x_z} \\
&= \frac{1}{2}\mathbf{x_z}^\top \left(\mathbf{u_z}^\top \mathbf{v_j}\right)\mathbf{x_j} + \frac{1}{2}\mathbf{x_j}^\top \left(\mathbf{u_j}^\top \mathbf{v_z}\right)\mathbf{x_z} + \frac{1}{2}\mathbf{x_j}^\top \left(\mathbf{u_j}^\top \mathbf{v_z}\right)\mathbf{x_z} + \frac{1}{2}\mathbf{x_z}^\top \left(\mathbf{u_z}^\top \mathbf{v_j}\right)\mathbf{x_j} \\
&= \mathbf{x_z}^\top \mathbf{u_z}^\top \mathbf{v_j}\mathbf{x_j} + \mathbf{x_j}^\top \mathbf{u_j}^\top \mathbf{v_z}\mathbf{x_z} \\
&= \mathbf{x_z}^\top B[z,j]\mathbf{x_j} + \mathbf{x_j}^\top B[j,z]\mathbf{x_z}
\end{aligned}$$

For each of c_{out} output channels, we get a matrix B^{sym} of dimension $k^2 n \times k^2 n$, giving $\mathbf{B}^{\text{sym}}$ a total shape of $[k^2 n, k^2 n, c_{out}]$.

B Training Parameters

Parameter	Type	Default Value	Description
Distribution Mode	String	`easy`	Difficulty or type of environment distribution. Choices: `easy`, `hard`, `exploration`, `memory`, `extreme`.
Environment Name	String	`maze`	Name of the environment to train on.
Number of Environments	Integer	64	Number of environments to use in parallel during training.
Number of Levels	Integer	100,000	Number of unique levels available for training.
Start Level	Integer	0	Starting level of the environment.
Method Label	String	`hazelnut`	Label or identifier for the method used.
GPU ID	Integer	7	GPU ID to use for training. Default is set to target GPU 7.
Learning Rate	Float	0.0001	Learning rate for the optimizer.
Entropy Coefficient	Float	0.01	Coefficient controlling entropy regularization.
Value Function Coefficient	Float	0.5	Coefficient balancing value function loss during training.
Discount Factor (γ)	Float	0.999	Discount factor for future rewards.
Lambda (λ)	Float	0.95	Generalized advantage estimation (GAE) discount factor.
Clip Range	Float	0.2	PPO clip range for policy loss updates.
Maximum Gradient Norm	Float	0.5	Maximum allowable gradient norm for clipping.
Steps per Update	Integer	256	Number of environment steps per policy update.
Batch Size	Integer	8	Batch size used for training.
Number of Epochs per Update	Integer	3	Number of training epochs per policy update.
Maximum Training Steps	Integer	12,800,000,000	Maximum number of total environment steps for training.
Pooling Method	String	`avg`	Pooling method used in the architecture. Options: `avg`, `max`, etc.

Table 1: Training parameters for ProcGen training

C Probe Results

Layer	Sequence 0 (%)	Sequence 1 (%)	Sequence 2 (%)	Fully Connected (%)
Initial Conv	99.88			-
Conv	100.00	100.00	100.00	-
MaxPool	99.88	100.00	100.00	-
ResBlock0	99.88	100.00	100.00	-
ResBlock0 Gated Conv	59.28	42.75	69.07	-
ResBlock1	100.00	100.00	100.00	-
ResBlock1 Gated Conv	69.07	0.00	40.12	-
Gated FC	-	-	-	68.36
Logits FC	-	-	-	81.20
Value FC	-	-	-	2.73

Table 2: F_1 scores for position probes trained on the output of different layers of the network. The scores indicate how well each layer preserves cheese position information.

D Training resources

Each ProcGen environment and model combination trained in 15-20 GPU hours on a 48GB VRAM 4xA40 gpu node

E Ablation supplemental figure

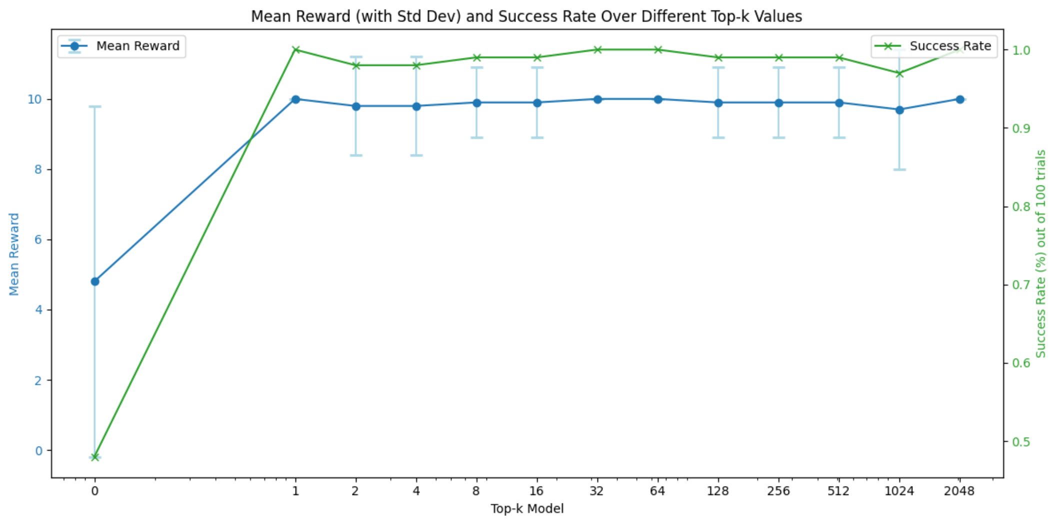

Figure 13: Keeping just first eigenvector for each output action in the final FC layer is enough to preserve near 100% success rate in solving mazes.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification:

   We make 4 claims in the contributions section of our Introduction.

   For (1), see Figure 1, Figure 2, and BIMPALA matches IMPALA performance

   For (2), see Decomposing Convolutions and Appendix A

   For (3), see Ablation within the standard bases and Figure 7

   For (4), see Protocol to enhance interpretability for RL with probes, Interpretability based on action features, Steering Experiments and corresponding figures in each subsection

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [YES]

   Justification: We discuss the paper's Limitations in the Discussion

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [YES]

   Justification: Our decomposition method does not make any assumptions other than a bilinear convolution, which is clearly stated in the text.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [YES]

   Justification: Our paper provides all the information necessary in order to reproduce our results, including traning parameters.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility.

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We acknowledge the value of open-source contributions for reproducibility. However, the current codebase requires significant refactoring to remove internal dependencies and improve documentation before public release. We plan to provide a clean, well-documented implementation post-review. The manuscript includes comprehensive methodological details and hyperparameter specifications to facilitate understanding and reimplementation.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We include a table of training parameters in Appendix B and discuss seeding and maze generation in Eigenfilter decomposition for singular probe channels

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Error bars as SEM, as stated in Figure 5's caption for throughout the paper, are plotted whenever appropriate. For training runs, we only ran each training once, and hence no variance is reported.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We describe the training resources in Appendix D

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read the code of ethics and we do not violate any of the points listed.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: While generally useful for future AI safety, the immediate impacts of our work on society are essentially nonexistent. Given the scope of the paper, we can not justify making claims towards a better, safer, society. Equally, nothing in our paper justifies safety risk concerns.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We don't upload a new model or any tools. Our model is simply a simplified form of an already existing model and we do not use it in any dangerous context.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: ProcGen and procgen-tools assets were used in compliance with the MIT License and cited in the paper. IMPALA was simplified and coded from scratch in pytorch in compliance with GNU license and cited in the paper.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We dont introduce new assets

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not crowdsource or use human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: No human or living subjects were used in the study.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: We did not use LLMs beyond getting feedback on certain, limited lines of code. The manuscript was written without LLM assistance.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.