Discovering and Steering Code Correctness through Residual Stream Directions in LLMs

Anonymous ACL submission

Abstract

Code-generating language models do not explicitly represent functional correctness: their token-level probabilities are only weakly correlated with the functional validity of generated programs, which can limit their reliability in high-stakes settings. Despite this, limited work has explored whether correctness is internally represented in model activations in a structured and causally usable form. In this work, we find that code correctness is well-approximated by a one-dimensional linear subspace in the residual stream of open-weight language models ranging from 1.5B to 7B parameters. Across multiple benchmark datasets, we demonstrate that this direction enables reliable causal intervention via activation steering; specifically, steering in the positive direction increases code success rates by up to 46.91%, whereas steering in the opposite direction or ablating the component leads to performance drops of up to 32.23%. We further observe that intervention effectiveness is not determined solely by linear separability, but is jointly predicted by probe-based discriminability and geometric alignment with the extracted direction. Overall, our results suggest that code correctness is represented along a geometrically structured direction in the residual stream and can be leveraged to improve the reliability and performance of code language models without additional fine-tuning. Code is available at https://anonymous.4open.science/r/emnlp_codellm-230D.

1 Introduction

From function synthesis to bug repair (Chen et al., 2021; Hui et al., 2024), large language models (LLMs) have demonstrated strong performance across software engineering tasks and are increasingly deployed in production environments (Li et al., 2024). However, assessing functional correctness through test-case execution remains a fundamental challenge (Chen et al., 2021; Liu et al., 2023a). Most LLMs generate incorrect code with the same fluency and confidence as correct code (Spiess et al., 2024; Guo et al., 2025), posing serious reliability risks in high-stakes domains where errors carry significant consequences.

LLMs generate code by modelling token distributions (Brown et al., 2020), yet these probabilities are often weakly correlated with code correctness (Spiess et al., 2024); the most probable completion frequently fails to satisfy functional adequacy requirements.

This has led to growing interest in mechanistic interpretability as a means of understanding model representations beyond next-token prediction statistics (Bereska and Gavves, 2024; Rai et al., 2024). Existing studies have demonstrated applications in reasoning (Nanda et al., 2023), safety mechanisms (Arditi et al., 2024), and truthfulness (Marks and Tegmark, 2023). However, limited work (Cencerrado et al., 2026) has studied whether models encode representations of code correctness or functional adequacy.

In this work, we show that code correctness is represented by a one-dimensional subspace in the residual stream of code-generating LLMs. Specifically, we use contrastive pairs of code runs (Panickssery et al., 2023; Zou et al., 2023) to identify a single difference-in-means direction (Belrose et al., 2023; Marks and Tegmark, 2023) that linearly separates functionally correct from incorrect generations. We then apply layer-wise probing (Alain and Bengio, 2017; Hewitt and Manning, 2019) to localise the layers where correctness information is most linearly accessible. Additionally, we define a correctness-axis Symmetry Score showing that the direction associated with recovering failed generations is geometrically aligned with the direction associated with corrupting correct generations. Finally, we perform three types of activation interventions: forward steering, backward steering, and direction ablation (Arditi et al., 2024) on the top se-

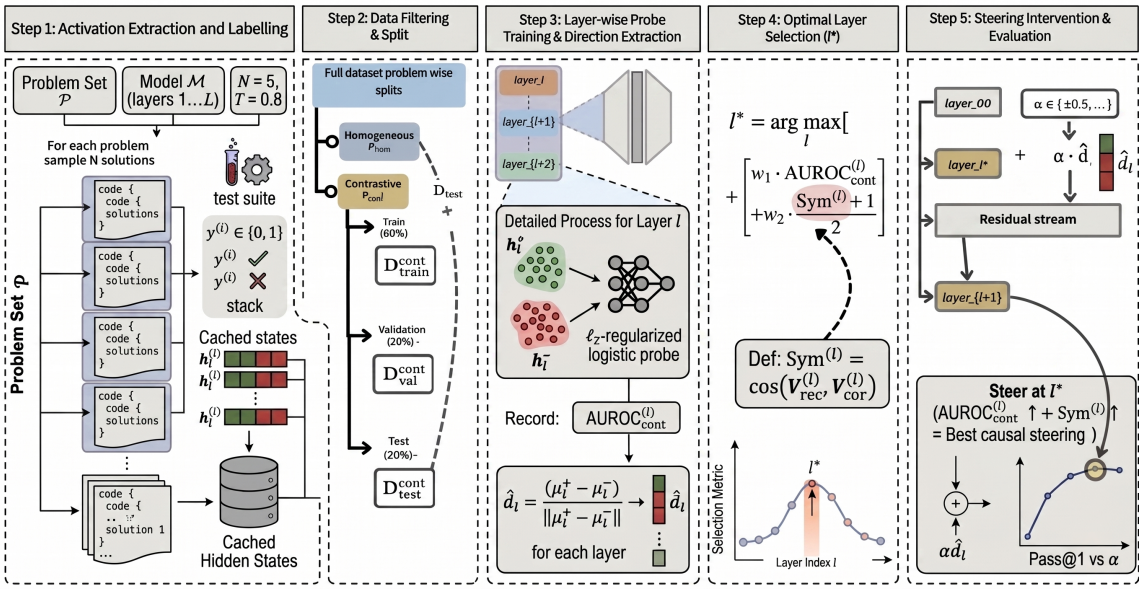

Figure 1: Overview of the five-stage pipeline: (1) activation extraction and labelling, (2) data filtering and splitting into homogeneous and contrastive subsets, (3) layer-wise logistic probe training and difference-in-means direction extraction, (4) optimal layer selection via a composite score combining AUROC and correctness-axis symmetry, and (5) residual-stream steering intervention and pass@1 evaluation.

lected layers to causally demonstrate bidirectional control and causal necessity of the correctness direction. Our contributions are as follows:

1. We identify a linear code-correctness direction in the residual stream of open-weight code LLMs and show that activation steering along this direction causally and bidirectionally shifts pass@1 performance.

2. We introduce a correctness-axis Symmetry Score and a composite layer-selection criterion that jointly capture probe discriminability and directional coherence for predicting steering effectiveness.

3. We open-source our evaluation framework spanning three code-generation benchmarks and two code language models to facilitate future research on interpretable and reliable code generation.

2 Related Work

The *linear representation hypothesis* holds that transformers encode semantic and factual properties as directions in activation space, recoverable via geometric operations on intermediate representations (Park et al., 2023; Marks and Tegmark, 2023; Burns et al., 2023). Probing classifiers formalise this by fitting lightweight linear models on layer-wise representations to recover structured properties (Burns et al., 2023), while the *difference-in-means* (DiM) estimator isolates behavioural directions truthfulness, refusal from contrastive activation pairs without gradient-based optimisation (Marks and Tegmark, 2023; Belrose et al., 2023; Arditi et al., 2024). Contrastive Activation Addition (CAA) (Panickssery et al., 2023) refines DiM by averaging over large contrastive datasets, yielding stable steering vectors for alignment-relevant behaviours. Activation and representation engineering then demonstrate that injecting these vectors into the residual stream at inference time reliably controls high-level output properties without any parameter update (Turner et al., 2023; Zou et al., 2023; Panickssery et al., 2023), with Arditi et al. (Arditi et al., 2024) showing refusal across 13 models collapses to a single DiM-recoverable direction. Recent work has systematically extended these methods through safety-perspective benchmarks of five canonical steering algorithms (Siu et al., 2025), gradient-free category-specific vectors (Ghosh et al., 2025), and contrastive personalisation of generation style (Zhang et al., 2025); broader surveys consolidate the field (Bereska and Gavves, 2024; Rai et al., 2024; Nanda et al., 2023).

Current interpretability research has yielded

Prompt:
Write a function to extract year, month and date from a url by using regex.

Generation (no intervention):

```
def extract_date(url):
    pattern = r'(\d{4})/(\d{1,})(1,2})'
    match = re.search(pattern, url)
    if match:
        return match.group(1), \
               match.group(2), \
               match.group(3)
    return None
```

✘ Returns ('2016', '04', '06') — strings not integers

Generation (with intervention):

```
import re
def extract_date(url):
    match = re.search(
        r'(\d{4}/(\d{2}/(\d{2})', url)
    if match:
        return int(match.group(1)), \
               int(match.group(2)), \
               int(match.group(3))
```

✓ Returns (2016, 4, 6) — correct integers

Figure 2: Adding the "code correctness" direction to residual stream activations enables recovery of failed runs into correct solutions. The example is from Qwen2.5-Code-7B-Instruct on the $\mathcal{D}_{\text{test}}$ pool of the MBPP+ dataset.

Prompt: Write a function to extract the index minimum value record from the given tuples.

Generation (no intervention):

```
def index_minimum(test_list):
    return min(test_list,
      key=lambda x: x[1])
```

✓ Returns ('Varsha', 100) — correct

Generation (intervention):

```
def index_minimum(test_list):
    return min(test_list,
      key=lambda x: x[0])
```

✘ Returns ('Goddess', 193) — sorts by name, not value

Figure 3: Ablating the "code correctness" direction in residual stream activations corrupts previously successful runs into failures. This example is taken from Qwen2.5-Code-1.5B-Instruct on the $\mathcal{D}_{\text{test}}$ pool of the MBPP+ dataset.

valuable insights through probing classifiers, attention analysis, and representational studies of model internals (Troshin and Chirkova, 2022; Paltenghi et al., 2024; Anand et al., 2024). Yet relatively little work has examined how correctness-related properties are encoded across layers (Cencerrado et al., 2026) offer an initial characterization of such cross-layer representations; open questions remain, however, regarding the joint geometry of multi-layer encodings and whether residual-stream interventions—analogous to those in the activation-steering literature—can improve generation reliability at inference time. Our work directly addresses this gap.

3 Methodology

In this section, we outline the procedure used to investigate whether a *linear direction* in the residual stream encodes code correctness and whether intervening this direction induces/reduces solution correctness during generation.

3.1 Dataset Filtering and Extraction

We partition the benchmark problems into two disjoint groups:

Contrastive Problems ($\mathcal{P}_{\text{cont}}$): Problems for which the sampled runs exhibit mixed correctness, i.e. containing least one correct and incorrect run.

$$\mathcal{P}_{\text{cont}} = \{\, p \mid \exists\, i, j \text{ s.t. } y^{(i)} \neq y^{(j)} \,\}. \quad (1)$$

Homogeneous Problems ($\mathcal{P}_{\text{hom}}$): Problems whose sampled runs are consistent in correctness, being either all correct or all incorrect.

$$\mathcal{P}_{\text{hom}} = \{\, p \mid \forall\, i, j,\ y^{(i)} = y^{(j)} \,\}. \quad (2)$$

For each benchmark problem, we generate N independent code completions (*runs*) using stochastic decoding with temperature $T = 0.8$. Each run is executed against the benchmark unit tests and assigned a binary correctness label based on execution outcome. Runs that pass all tests are labeled *correct*, while runs that fail at least one test are labeled *incorrect*. For each run, activations at the final generated token are recorded across all layers.

3.2 Probing

We use a logistic probe to predict code correctness from frozen intermediate representations, it implies that correctness is *linearly encoded* in the residual stream rather than entangled in a higher-order, non-linear structure. For each layer l, we train a probe on hidden states $h_l^{(i)} \in \mathbb{R}^d$ extracted at the final generated token of each solution:

$$\text{Probe}_l : h_l^{(i)} \hat{y}^{(i)} \in \{0, 1\}. \quad (3)$$

Each probe is trained with ℓ_2 regularization and balanced class weights, and evaluated using AUROC on the held-out contrastive validation split to quantify linear separability while remaining robust to class imbalance.

3.3 Extracting a Code Correctness Direction

Prior work has shown that semantic features can often be represented as approximately linear directions in transformer activation spaces (Park et al., 2023; Marks and Tegmark, 2023). To identify a *code correctness* direction in the residual stream, we employ the difference-in-means approach (Belrose et al., 2023). For each selected layer l, we compute the mean hidden-state activations for correct and incorrect runs separately:

$$\hat{d}_l = \frac{\mu^+ - \mu^-}{\|\mu^+ - \mu^-\|}, \qquad \mu^\pm = \mathrm{mean}(H_l^\pm), \quad (4)$$

where H_l^+ and H_l^- denote the sets of hidden states at layer l corresponding to correct and incorrect runs, respectively, extracted from $\mathcal{D}_{\text{cont}}^{\text{train}}$. The resulting vector $\hat{d}_l$ defines a layer-specific correctness direction in the residual stream activation space.

3.4 Correctness Axis Direction Symmetry

To verify that the code-correctness direction ($\hat{d}_l$) provides bidirectional causal control, we test whether the recovery direction (correcting failed runs) is geometrically aligned with the corruption direction obtained by steering in the opposite direction (inducing failures in successful runs). For each layer l, test problems from $\mathcal{P}_{\text{const}}$ are partitioned by difficulty:

- *Hard* ($\mathcal{H}$): $1 \leq n_{\text{passed}} \leq 2$ where the model rarely succeeds.
- *Easy*($\mathcal{E}$): $3 \leq n_{\text{passed}} \leq 4$ where the model mostly succeeds.

Let H_l^+ and H_l^- denote hidden states from correct and incorrect runs at layer l, respectively. We define two directional vectors:

$$V_{\text{rec}}^{(l)} = \mathrm{mean}\big(H_l^+[\mathcal{H}]\big) - \mathrm{mean}\big(H_l^-[\mathcal{H}]\big), \quad (5)$$

$$V_{\text{cor}}^{(l)} = \mathrm{mean}\big(H_l^-[\mathcal{E}]\big) - \mathrm{mean}\big(H_l^+[\mathcal{E}]\big). \quad (6)$$

$V_{\text{rec}}^{(l)}$ captures the transition from incorrect to correct activations on hard problems, corresponding to recovery of failing runs. Conversely, $V_{\text{cor}}^{(l)}$ captures the transition from correct to incorrect activations on easy problems, corresponding to corruption of successful runs.

Symmetry Score: To quantify the extent to which recovery and corruption share a common geometric axis for layer l, we compute $\mathrm{Sym}^{(l)}$ defined as the cosine similarity between $V_{\text{rec}}^{(l)}$ and the negated corruption direction $-V_{\text{cor}}^{(l)}$:

$$\mathrm{Sym}^{(l)} = \frac{-V_{\text{rec}}^{(l)} \cdot V_{\text{cor}}^{(l)}}{\|V_{\text{rec}}^{(l)}\| \, \|V_{\text{cor}}^{(l)}\|}. \quad (7)$$

A score near +1 indicates a unified correctness axis underlying both recovery and corruption directions, whereas scores near 0 or below suggest distinct geometric structures that limit the effectiveness of single-direction steering.

3.5 Intervention Layer Selection

We select the intervention layers through a composite score l^*_{score} that combines probe discriminability and correctness-axis symmetry defined as:

$$l^*_{\text{score}} = \arg\max_l \left[w_1 A^{(l)} + w_2 \tilde{S}^{(l)} \right] \quad (8)$$

where $A^{(l)}$ represents the AUROC score, and $\tilde{S}^{(l)} = \frac{\mathrm{Sym}^{(l)}+1}{2}$ denotes the normalized symmetry score at layer l.

The normalisation $(\mathrm{Sym}^{(l)} + 1)/2$ maps symmetry scores from $[-1, 1]$ to $[0, 1]$, enabling direct comparability with AUROC. We compute l^*_{score} for the candidate layers ranked by $\mathrm{AUROC}^{(l)}$ and then re-rank by the composite score to select the best intervention layers. We set the weighting coefficients to $w_1 = 0.6$ and $w_2 = 0.4$ in our experiments, balancing probe discriminability and symmetry respectively.

3.6 Model Intervention

For a candidate layer l with direction $\hat{d}_l$, we regulate the strength of the corresponding correctness feature through controlled linear interventions applied to the residual stream during autoregressive generation. For this study, we perform three types of interventions defined as follows:

Forward Steering We add a scaled correctness direction to the residual-stream activations of failed runs to shift them towards to the mean activation of correct runs (μ^+) as given below:

$$h_l \leftarrow h_l + \alpha \cdot \hat{d}_l, \quad \alpha > 0. \quad (9)$$

Backward Steering Similarily, we add the attenuated correctness direction to the residual-stream activations of successful runs to corrupt and steer them towards to the mean activation of incorrect runs (μ^-) as mentioned below:

$$h_l \leftarrow h_l + \alpha \cdot \hat{d}_l, \quad \alpha < 0. \quad (10)$$

Direction Ablation Following Arditi et al. (2024), we evaluate the causal necessity of the learned correctness direction by removing its aligned component from the residual-stream activations as defined below:

$$h_l \leftarrow h_l - \alpha \cdot (h \cdot \hat{d}_l) \cdot \hat{d}_l, \quad \alpha = 1. \quad (11)$$

4 Experimental Setup

This section outlines the datasets, models, evaluation metrics and the experimental implementation used in this study.

4.1 Datasets

MBPP+ (Austin et al., 2021) and *HumanEval+* (Liu et al., 2023b) are two execution-based Python code-generation benchmarks both introduced by EvalPlus (Liu et al., 2023c). *MBPP+* extends the `MBPP` (Mostly Basic Python Problems) benchmark with 399 programming tasks and up to $35\times$ more test cases, while *HumanEval+* extends the `HumanEval` benchmark with 164 handwritten function-level programming problems and up to $80\times$ expanded test coverage.

BigCodeBench (Zhuo et al., 2025) is an Python code-generation benchmark consisiting of 1140 instruction-driven programming problems covering algorithmic reasoning, software engineering, and library-intensive code generation. For this study, we use the `BigCodeBench-Complete` variant with `v0.1.4` split for benchmarking our approach.

4.2 Models

We conduct experiments using two open-weight instruction-tuned code-generation models from the Qwen2.5-Coder family (Hui et al., 2024): Qwen2.5-Coder-1.5B-Instruct (1.5B parameters, 28 layers) and Qwen2.5-Coder-7B-Instruct (7B parameters, 28 layers). For brevity, we refer to these models as Q-1.5B and Q-7B respectively.

4.3 Evaluation Metrics

pass@1: This denotes the percentage of benchmark problems for which a single generated run/solution passes all test cases.

$$\text{p@1} = \frac{100}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \mathbf{1}_p \tag{12}$$

Intervention Δpass@1: To evaluate the causal role of the discovered directions, we compute the percentage-point change in pass@1 between the steered and baseline models across specified subsets of runs.

$$\Delta\text{p@1}_d = \text{p@1}_{\text{steered}} - \text{p@1}_{\text{base}} \tag{13}$$

where $d \in \{\text{fwd}, \text{bwd}, \text{abl}\}$.

For forward steering ($d = \text{fwd}$), the metric is computed over baseline *failed runs*, where a positive value indicates successful recovery. For both backward steering ($d = \text{bwd}$) and direction ablation ($d = \text{abl}$), the metric is computed over baseline *successful runs*. In these cases, a negative value indicates performance degradation, reflecting targeted corruption (backward steering) and the causal necessity of the direction (ablation), respectively.

4.4 Implementation Details

The pipeline proceeds in four stages. In the *activation extraction* stage, each model generates $N = 5$ solutions per problem at temperature $T = 0.8$, using a different random seed for every run. This sampling induces behavioral diversity, producing both successful and failed trajectories across problems and enabling reliable estimation of internal representations of correctness. For every run, a full forward pass is executed and the residual-stream activations at the final generated token are recorded across all layers and stored in HDF5 format. Solutions are subsequently evaluated on all test cases to assign binary correctness labels.

Next in the *dataset filtering and splitting* stage, we first filter the dataset problems into 2 subsets namely Homogenous ($\mathcal{P}_{\text{how}}$) and Contrastive ($\mathcal{P}_{\text{cont}}$. We then partition $\mathcal{P}_{\text{cont}}$ into disjoint train, validation, and test splits at the problem level using a $60/20/20$ ratio. The training split $\mathcal{D}^{\text{cont}}_{\text{train}}$ is used for linear probe training and correctness-direction extraction, while the validation split $\mathcal{D}^{\text{cont}}_{\text{val}}$ is used for probe evaluation, layer selection, and correctness-axis symmetry analysis. Finally, steering experiments are performed on a held-out evaluation pool $\mathcal{D}_{\text{test}}$ consisting of $\mathcal{D}^{\text{cont}}_{\text{test}}$ together with $\mathcal{P}_{\text{hom}}$, thereby ensuring that intervention performance is evaluated over the full distribution of benchmark behaviours rather than only on contrastive problems.s evaluated over the full distribution of benchmark behaviours rather than only on contrastive problems.

Further, in the *probing* stage, we train a logistic probe for each layer with ℓ_2 regularization, and rank layers according to l^*_{score}, selecting the Top-K (K=3) layers for intervention. Finally, in the *steering* stage, we load pre-computed directions for the top-3 layers and apply interventions on fresh generations using residual-stream hooks over $\mathcal{D}_{\text{test}}$. Forward steering is applied to incorrect runs, while backward steering and direction ablation are applied to correct runs. We further sweep steering coefficients over $\alpha \in \{\pm 0.5, \pm 1.0, \pm 2.0, \pm 5.0, \pm 10.0\}$ in both directions. All experiments are conducted over 5 in-

	Δp@1$_{\mathbf{fwd}}$(↑)					Δp@1$_{\mathbf{bwd}}$(↓)				
Layer	α=**0.5**	**1**	**2**	**5**	**10**	α=**−0.5**	**−1**	**−2**	**−5**	**−10**
Q-1.5B										
L-06	$+24.60_{(0.64)}$	$\mathbf{+24.82_{(1.39)}}$	$+24.17_{(0.65)}$	$+23.78_{(2.01)}$	$+24.74_{(1.09)}$	$-9.48_{(3.07)}$	$-11.06_{(1.47)}$	$-11.91_{(1.77)}$	$\mathbf{-12.33_{(2.26)}}$	$-11.47_{(1.47)}$
L-08	$+24.42_{(1.42)}$	$\mathbf{+27.92_{(0.37)}}$	$+27.21_{(1.09)}$	$+26.81_{(2.24)}$	$+26.52_{(1.33)}$	$-10.06_{(0.81)}$	$-10.20_{(0.41)}$	$-10.11_{(1.47)}$	$-10.62_{(3.07)}$	$\mathbf{-11.01_{(4.40)}}$
L-19	$+26.60_{(1.18)}$	$+27.39_{(0.68)}$	$+27.95_{(1.07)}$	$\mathbf{+28.02_{(0.32)}}$	$+26.28_{(1.33)}$	$-13.33_{(0.41)}$	$-13.06_{(3.54)}$	$-14.05_{(0.41)}$	$-14.02_{(1.41)}$	$\mathbf{-14.21_{(4.62)}}$
Q-7B										
L-19	$+45.38_{(0.23)}$	$+44.63_{(0.48)}$	$+46.19_{(1.13)}$	$\mathbf{+46.91_{(1.26)}}$	$+45.88_{(0.13)}$	$-14.22_{(1.44)}$	$-13.29_{(1.51)}$	$-13.99_{(1.51)}$	$-14.22_{(1.65)}$	$\mathbf{-16.32_{(4.01)}}$
L-20	$+44.44_{(1.34)}$	$+45.38_{(0.79)}$	$+44.44_{(0.58)}$	$\mathbf{+45.92_{(1.15)}}$	$+45.16_{(1.21)}$	$-13.99_{(2.06)}$	$-13.91_{(1.51)}$	$\mathbf{-14.69_{(0.99)}}$	$-14.42_{(2.06)}$	$-15.85_{(3.72)}$
L-22	$+44.91_{(0.13)}$	$+44.07_{(1.26)}$	$\mathbf{+45.72_{(0.35)}}$	$+45.26_{(1.00)}$	$+44.65_{(1.69)}$	$-13.75_{(1.84)}$	$-12.82_{(1.19)}$	$-13.93_{(0.33)}$	$\mathbf{-14.45_{(1.44)}}$	$-12.82_{(1.19)}$

Table 1: Forward and backward steering (Δp@1$_{fwd/bwd}$) on MBPP+ across the top-3 selected layers and all steering strengths α, for both evaluated models. Results are reported as mean $\pm$ standard deviation, with bold indicating the best-performing α per layer.

	Δp@1$_{\mathbf{fwd}}$(↑)					Δp@1$_{\mathbf{bwd}}$(↓)				
Layer	α=**0.5**	**1**	**2**	**5**	**10**	α=**−0.5**	**−1**	**−2**	**−5**	**−10**
Q-1.5B										
L-15	$+24.10_{(1.92)}$	$+20.77_{(0.63)}$	$\mathbf{+24.62_{(1.09)}}$	$+22.31_{(2.74)}$	$+16.15_{(1.26)}$	$-14.17_{(4.71)}$	$-15.17_{(6.56)}$	$-15.00_{(4.08)}$	$-13.33_{(8.25)}$	$\mathbf{-16.17_{(1.18)}}$
L-24	$+22.31_{(1.09)}$	$+21.03_{(1.58)}$	$+22.56_{(1.31)}$	$\mathbf{+23.08_{(1.09)}}$	$+20.51_{(2.83)}$	$\mathbf{-17.50_{(4.08)}}$	$-15.83_{(5.89)}$	$-14.17_{(5.14)}$	$-15.83_{(5.14)}$	$-13.33_{(5.14)}$
L-25	$\mathbf{+22.05_{(0.73)}}$	$+20.51_{(1.31)}$	$+20.51_{(0.96)}$	$+22.05_{(1.45)}$	$+21.54_{(1.66)}$	$-17.50_{(2.40)}$	$\mathbf{-17.50_{(1.40)}}$	$-15.83_{(2.25)}$	$-15.83_{(2.25)}$	$-14.17_{(1.25)}$
Q-7B										
L-19	$+40.00_{(3.93)}$	$+40.89_{(4.40)}$	$+42.67_{(3.93)}$	$\mathbf{+43.11_{(2.74)}}$	$+41.78_{(0.63)}$	$-26.67_{(1.31)}$	$\mathbf{-29.12_{(1.79)}}$	$-28.42_{(0.86)}$	$-27.72_{(1.31)}$	$-29.12_{(3.88)}$
L-20	$+42.22_{(2.74)}$	$\mathbf{+42.67_{(2.88)}}$	$+40.89_{(2.74)}$	$+39.56_{(1.66)}$	$+41.33_{(3.27)}$	$-27.37_{(1.72)}$	$-27.72_{(0.50)}$	$-26.67_{(1.79)}$	$-28.77_{(1.79)}$	$\mathbf{-31.23_{(1.79)}}$
L-22	$+40.00_{(3.27)}$	$\mathbf{+43.56_{(3.50)}}$	$+42.22_{(3.14)}$	$+39.11_{(4.53)}$	$+41.78_{(3.14)}$	$-25.96_{(1.31)}$	$-25.96_{(0.50)}$	$-27.37_{(0.86)}$	$-27.72_{(0.99)}$	$\mathbf{-28.42_{(2.27)}}$

Table 2: Forward and backward steering (Δp@1$_{fwd/bwd}$) on HumanEval+ across the top-3 selected layers and all steering strengths α, for both evaluated models. Results are reported as mean $\pm$ standard deviation, with bold indicating the best-performing α per layer.

dependent trials, with all metrics reported as mean $\pm$ standard deviation across trials. We load model weights from HuggingFace in `bfloat16` precision and train using a batch size of 32 on a single RTX A5000 (24GB VRAM).

5 Results

On MBPP+ (Tables 1 and 4), for Q-1.5B the best layer is L-19 ($l^*_{\text{score}} = 0.782$), achieving a forward gain of $+28.02\%$ at $\alpha = 5$, a backward degradation of -14.21% at $\alpha = -10$, and a direction ablation of -36.21%, while the worst layer L-06 ($l^*_{\text{score}} = 0.686$) yields only +24.82% forward, -12.33% backward, and -35.34% ablation ($\pm$2.11); the higher AUROC and symmetry at L-19 jointly produce a more precise correctness direction, translating directly into stronger and more stable intervention effects across all three modes. For Q-7B, the best layer is L-20 ($l^*_{\text{score}} = 0.870$), achieving +45.92% forward at $\alpha = 5$, -14.69% backward at $\alpha = -2$, and -13.99% ablation, whereas the worst layer L-19 ($l^*_{\text{score}} = 0.786$) yields +46.91% forward but only -13.99% ablation with higher variance, its lower symmetry score despite competitive AUROC confirming that geometric coherence is a necessary complement to linear discriminability for reliable bidirectional control.

On HumanEval+ (Tables 2 and 5), for Q-1.5B the best layer is L-24 ($l^*_{\text{score}} = 0.790$), achieving +23.08% forward at $\alpha = 5$, -17.50% backward at $\alpha = -0.5$, and -37.50% ablation, while the worst layer L-15 ($l^*_{\text{score}} = 0.766$) yields only +24.62% forward, -16.17% backward, and a large but high-variance ablation of -41.67% ($\pm$1.18); the lower AUROC at L-15 introduces noise into the extracted direction, producing broad residual-stream disruption under ablation without enabling comparably reliable additive steering. For Q-7B, the best layer is L-22 ($l^*_{\text{score}} = 0.810$), achieving +43.56% forward at $\alpha = 1$, -28.42% backward at $\alpha = -10$, and -25.96% ablation, whereas the worst layer L-20 ($l^*_{\text{score}} = 0.778$) yields +42.67% forward, -31.23% backward, and -24.91% ablation; the reduced symmetry at L-20 relative to L-22 weakens the geometric alignment of the correctness axis, limiting the completeness of directional removal under ablation despite its competitive forward steering gain.

On BigCodeBench (Tables 3 and 6), for Q-1.5B the best layer is L-17 ($l^*_{\text{score}} = 0.766$), achieving +7.51% forward at $\alpha = 5$, -7.12% backward at $\alpha = -10$, and -60.89% ablation, while the worst layer L-19 ($l^*_{\text{score}} = 0.720$) yields only +6.85% forward, -7.89% backward, and -62.54% ablation; although L-19 produces the largest ablation degra-

Layer	Δp@1$_{\mathbf{fwd}}$(↑)					Δp@1$_{\mathbf{bwd}}$(↓)				
	α=**0.5**	**1**	**2**	**5**	**10**	α=**−0.5**	**−1**	**−2**	**−5**	**−10**
Q-1.5B										
L-17	$+6.38_{(0.45)}$	$+6.75_{(0.38)}$	$+6.35_{(0.82)}$	$\mathbf{+7.51_{(0.52)}}$	$+6.91_{(0.12)}$	$-6.95_{(3.51)}$	$-6.25_{(4.65)}$	$-6.28_{(4.61)}$	$-7.12_{(3.17)}$	$\mathbf{-7.12_{(1.45)}}$
L-18	$+6.08_{(0.50)}$	$+6.25_{(0.48)}$	$\mathbf{+6.85_{(0.33)}}$	$+6.28_{(0.59)}$	$+6.19_{(0.31)}$	$-4.66_{(3.51)}$	$-6.42_{(4.34)}$	$-7.36_{(3.17)}$	$\mathbf{-7.60_{(2.89)}}$	$-7.06_{(1.00)}$
L-19	$+6.65_{(0.47)}$	$+6.41_{(0.26)}$	$+6.56_{(0.19)}$	$+6.05_{(0.12)}$	$\mathbf{+6.85_{(0.29)}}$	$-6.25_{(2.59)}$	$-7.19_{(5.34)}$	$\mathbf{-7.89_{(3.27)}}$	$-7.36_{(1.66)}$	$-7.18_{(2.59)}$
Q-7B										
L-19	$+15.84_{(1.53)}$	$+15.97_{(1.12)}$	$+15.98_{(2.72)}$	$+16.35_{(1.07)}$	$\mathbf{+16.48_{(1.21)}}$	$-12.70_{(0.31)}$	$-13.28_{(2.01)}$	$-12.42_{(1.58)}$	$-13.56_{(1.09)}$	$\mathbf{-13.68_{(2.41)}}$
L-20	$+16.34_{(0.48)}$	$\mathbf{+16.59_{(1.02)}}$	$+15.81_{(1.24)}$	$+15.46_{(1.02)}$	$+15.48_{(1.06)}$	$-11.85_{(2.06)}$	$-11.91_{(1.11)}$	$-12.28_{(0.93)}$	$\mathbf{-12.91_{(3.51)}}$	$-12.13_{(1.14)}$
L-22	$+15.46_{(1.32)}$	$+15.72_{(1.10)}$	$\mathbf{+16.60_{(0.81)}}$	$+15.34_{(1.07)}$	$+16.10_{(2.05)}$	$-12.71_{(1.07)}$	$-12.71_{(0.82)}$	$-12.99_{(1.04)}$	$-11.73_{(0.52)}$	$\mathbf{-12.99_{(1.04)}}$

Table 3: Forward and backward steering (Δp@1$_{fwd/bwd}$) on BigCodeBench across the top-3 selected layers and all steering strengths α, for both evaluated models. Results are reported as mean $\pm$ standard deviation, with bold indicating the best-performing α per layer.

Model	Layer	AUROC$^{(l)}$	Sym$^{(l)}$	l^*_{score}
Q-1.5B	L-06	0.63	0.54	0.686
	L-08	0.75	0.72	**0.794**
	L-19	0.75	0.66	0.782
Q-7B	L-19	0.84	0.41	0.786
	L-20	0.89	0.68	**0.870**
	L-22	0.86	0.51	0.818

Table 4: Top-K (K=3) layer selection per model on MBPP+, sorted by layer depth ascending.

Model	Layer	AUROC$^{(l)}$	Sym$^{(l)}$	l^*_{score}
Q-1.5B	L-15	0.74	0.62	0.766
	L-24	0.77	0.64	**0.790**
	L-25	0.76	0.66	0.788
Q-7B	L-19	0.81	0.61	0.808
	L-20	0.79	0.51	0.778
	L-22	0.83	0.56	**0.810**

Table 5: Top-K (K=3) layer selection per model on HumanEval+, sorted by layer depth ascending.

dation, its low AUROC of 0.64 reflects a noisy correctness direction that causes broad disruption under ablation without translating into reliable additive steering, a dissociation that confirms linear discriminability must accompany geometric coherence for precise causal control. For Q-7B, the best layer is L-22 ($l^*_{\text{score}} = 0.786$), achieving +16.60% forward at $\alpha = 2$, -12.99% backward at $\alpha = -2$, and -52.99% ablation, whereas the worst layer L-19 ($l^*_{\text{score}} = 0.782$) yields +16.48% forward, -13.68% backward, and -52.42% ablation; across all three benchmarks, the monotonic alignment between l^*_{score} and intervention magnitude consistently holds, confirming that jointly high linear discriminability and directional coherence are necessary conditions for effective and stable activation-based control of code correctness.

Model	Layer	AUROC$^{(l)}$	Sym$^{(l)}$	l^*_{score}
Q-1.5B	L-17	0.68	0.79	**0.766**
	L-18	0.66	0.73	0.742
	L-19	0.64	0.68	0.720
Q-7B	L-19	0.73	0.72	0.782
	L-20	0.74	0.65	0.774
	L-22	0.77	0.62	**0.786**

Table 6: Top-K (K=3) layer selection per model on BigCodeBench, sorted by layer depth ascending.

6 Conclusion

In this paper, we show that code correctness is encoded as a linear direction in the internal representations of code LLMs, indicating that these models develop a notion of functional adequacy beyond surface-level output probabilities. Across three coding benchmarks and two LLMs, we demonstrate that this direction enables bidirectional causal control via activation steering and is causally necessary, as confirmed by direction ablation. We further observe that a layer's linear discriminability and its geometric alignment with the correctness axis jointly act as strong predictors of steering effectiveness. More broadly, these findings suggest that mechanistic representations of correctness can be systematically identified and used to interpret and control model behaviour in code generation systems.

Limitations

Although our study provides valuable insights into the mechanistic representations underlying code generation, it has several limitations. First, the reliance on logistic probes may limit the ability to

Dataset	Model	Layer	Δp@1$_{abl}$ (↓)
MBPP+	Q-1.5B	L-06	$-35.34_{(2.11)}$
		L-08	$-33.33_{(0.81)}$
		L-19	$\mathbf{-36.21_{(2.11)}}$
	Q-7B	L-19	$\mathbf{-13.99_{(2.85)}}$
		L-20	$-13.99_{(2.97)}$
		L-22	$-13.29_{(0.99)}$
HumanEval	Q-1.5B	L-15	$\mathbf{-41.67_{(1.18)}}$
		L-24	$-37.50_{(3.54)}$
		L-25	$-41.67_{(2.36)}$
	Q-7B	L-19	$\mathbf{-27.02_{(2.76)}}$
		L-20	$-24.91_{(1.98)}$
		L-22	$-25.96_{(1.79)}$
BigCodeBench	Q-1.5B	L-17	$-60.89_{(3.74)}$
		L-18	$-62.54_{(3.04)}$
		L-19	$\mathbf{-62.54_{(6.32)}}$
	Q-7B	L-19	$-52.42_{(0.00)}$
		L-20	$\mathbf{-53.56_{(0.00)}}$
		L-22	$-52.99_{(0.00)}$

Table 7: Direction ablation (Δp@1$_{abl}$) on MBPP+, HumanEval+, and BigCodeBench across the top-3 selected layers for both models.

capture more complex, non-linear structures of correctness encoded in the hidden activations. Second, the procedure used to derive the code correctness vector may not be optimal. This work primarily serves as a proof for existence of such a direction rather than providing a comprehensive investigation of optimal extraction methods. Finally, our experiments are conducted on two models within the 7B parameter range due to hardware constraints. As a result, the generalizability of our findings to other models, particularly larger-scale state-of-the-art proprietary systems and future architectures, remains uncertain.

References

Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes. *International Conference on Learning Representations Workshop*.

Abhinav Anand, Shweta Verma, Krishna Narasimhan, and Mira Mezini. 2024. A critical study of what code-llms (do not) learn. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15869–15889.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Rimsky, Wes Gurnee, and Neel Nanda. 2024. Refusal in language models is mediated by a single direction. *Advances in Neural Information Processing Systems*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor McKinney, Lev Ostrowski, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.

Leonard Bereska and Efstratios Gavves. 2024. Mechanistic interpretability for AI safety — a review. *Transactions on Machine Learning Research*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2023. Discovering latent knowledge in language models without supervision. *International Conference on Learning Representations*.

Xavier Cencerrado and 1 others. 2026. Representations of code correctness in large language models. *arXiv preprint arXiv:2601.04112*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Shaona Ghosh, Amrita Bhattacharjee, Yftah Ziser, and Christopher Parisien. 2025. Safesteer: Interpretable safety steering with refusal-evasion in llms. *arXiv preprint arXiv:2506.04250*.

Yifeng Guo and 1 others. 2025. Do LLMs know when their code is buggy? exploring code correctness self-awareness. *arXiv preprint arXiv:2502.09798*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. *Proceedings of NAACL*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, and 1 others. 2024. Qwen2.5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Zhenyang Li and 1 others. 2024. Measuring the impact of AI coding tools on developer productivity. *arXiv preprint arXiv:2410.12944*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023a. EvalPlus: Rigorously evaluating LLM-Synthesized code. *Advances in Neural Information Processing Systems*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023b. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023c. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. *Thirty-seventh Conference on Neural Information Processing Systems*.

Samuel Marks and Max Tegmark. 2023. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *arXiv preprint arXiv:2310.06824*.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. *International Conference on Learning Representations*.

Matteo Paltenghi, Rahul Pandita, Austin Z Henley, and Albert Ziegler. 2024. Follow-up attention: An empirical study of developer and neural model code exploration. *IEEE Transactions on Software Engineering*, 50(10):2568–2582.

Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alex Turner. 2023. Steering LLaMA 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*.

Kiho Park, Yo Joong Choe, and Victor Veitch. 2023. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*.

Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. 2024. A practical review of mechanistic interpretability for transformer-based language models. *arXiv preprint arXiv:2407.02646*.

Vincent Siu, Nicholas Crispino, David Park, Nathan W. Henry, Zhun Wang, Yang Liu, Dawn Song, and Chenguang Wang. 2025. Steeringsafety: A systematic safety evaluation framework of representation steering in llms. *arXiv preprint arXiv:2509.13450*.

Jannik Spiess and 1 others. 2024. Quality over quantity: Boosting LLM performance with self-directed data curation via V-usable information. *arXiv preprint arXiv:2406.11271*.

Sergey Troshin and Nadezhda Chirkova. 2022. Probing pretrained models of source codes. In *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 371–383.

Alexander Matt Turner, Lisa Thiergart, Gavin Udell, David Leech, Ulisse Mini, and Monte MacDiarmid. 2023. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*.

Jinghao Zhang, Yuting Liu, Wenjie Wang, Qiang Liu, Shu Wu, Liang Wang, and Tat-Seng Chua. 2025. Personalized text generation with contrastive activation steering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7128–7141.

Terry Yue Zhuo, Vu Minh Chien, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, and 1 others. 2025. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. *The Thirteenth International Conference on Learning Representations*.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, and 1 others. 2023. Representation engineering: A top-down approach to AI transparency. *arXiv preprint arXiv:2310.01405*.