Understanding Game-Playing Agents with Natural Language Annotations

Nicholas Tomlin **Andre He** **Dan Klein**
Computer Science Division, University of California, Berkeley
{nicholas_tomlin, andre.he, klein}@berkeley.edu

Abstract

We present a new dataset containing 10K human-annotated games of Go and show how these natural language annotations can be used as a tool for model interpretability. Given a board state and its associated comment, our approach uses linear probing to predict mentions of domain-specific terms (e.g., *ko*, *atari*) from the intermediate state representations of game-playing agents like AlphaGo Zero. We find these game concepts are nontrivially encoded in two distinct policy networks, one trained via imitation learning and another trained via reinforcement learning. Furthermore, mentions of domain-specific terms are most easily predicted from the later layers of both models, suggesting that these policy networks encode high-level abstractions similar to those used in the natural language annotations.

1 Introduction

Go is fundamentally a game of pattern recognition: from *ladders* and *walls* to *sente* and *shape*, professional players rely on a rich set of concepts to communicate about structures on the game board. Some patterns are relatively simple: *walls* are lines of adjacent stones, and an *atari* is a threat to capture stones on the next move; other patterns are less clearly defined: *hane* refers to any move that "goes around" the opponent's stones, and *sente* describes a general state of influence or tempo. Despite the nebulous definitions of some of these terms, human players use them productively. Beginners learn about *eyes* that determine when groups of stones are *alive* or *dead* and are given guidelines for when they should play a *cut* or extend a *ladder*; more advanced players learn sequences of *joseki* and *tesuji* and are taught to distinguish *good shape* from *bad shape*. Figures 1-2 depict some example concepts.

Computers have recently surpassed human performance at Go (Silver et al., 2016), but relatively little is known about why these programs perform so well and whether they rely on similar representational units to choose the moves they play. While post-hoc behavioral analyses suggest that AlphaGo and its successor AlphaGo Zero (Silver et al., 2017) can process complex game situations involving *shape*, *capturing races*, *sente*, *tesuji*, and even *ladders*, existing interpretability work has focused on the moves that agents play, rather than the internal computations responsible for those moves. Our work instead proposes a *structural* analysis by correlating the internal representations of game-playing agents with information from a naturally-occurring dataset of move-by-move annotations.

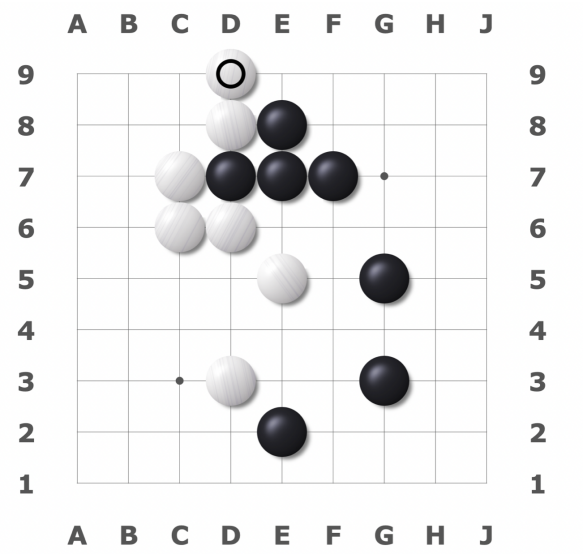
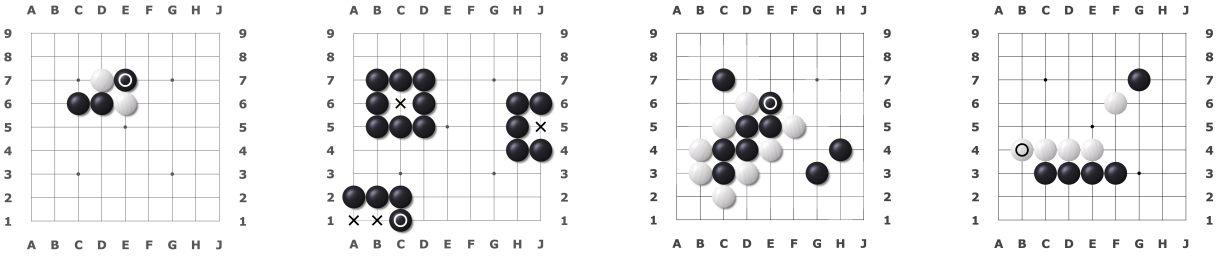


Figure 1: Example comment from our dataset, with domain-specific keywords (*shape*, *sente*) highlighted. Although this comment is from a 9 × 9 game for illustrative purposes, our dataset primarily focuses on annotations from 19 × 19 games.

In this paper, we use linear probing to explore how domain-specific concepts are represented by



(a) Cut (b) Eyes (c) Ladder (d) Wall

Figure 2: Example Go patterns and associated terminology. (a) *Cuts* are moves that separate two groups of stones. (b) *Eyes* are empty squares surrounded by stones of the same color. (c) *Ladders* are capturing races which may span the entire board. (d) Walls are lines of adjacent stones of the same color. These terms appear frequently in our dataset of natural language annotations and are further defined in the appendix.

game-playing agents. Because we do not have ground-truth labels explaining which concepts are relevant to a given game state, we collect a dataset of 10K annotated Go games (§2.1). Given a board state and its associated comment, we produce binary feature vectors summarizing which game phenomena (e.g., *ko*, *atari*) are mentioned in the comment and use pattern-based feature extractors to determine which phenomena are actually present on the board (§2.2). We then feed board states into two policy networks with disparate architectures and training methods (§3.1) to obtain intermediate representations. Finally, we use linear probes (§3.2) to predict the binary feature vectors from our policy networks. Generally, we find that pattern-based features are encoded in the early layers of policy networks, while natural language features are most easily extracted from the later layers of both models. We release our code and data at https://github.com/andrehe02/go.

2 Dataset

2.1 Annotated Games

We collect 10K games with move-by-move English annotations from the Go Teaching Ladder (GTL).[1] The GTL was created by Jean-loup Gailly and Bill Hosken in 1994 and maintained until 2016 and permits non-commercial digital redistribution. Until 2016, members of the GTL could submit games for review by volunteers, who ranged from amateur to professional strength. Reviewers were given annotation guidelines and required to have a higher rating than their assigned reviewees, resulting in high quality natural language data. Of the collected games, we focus on 9524 which were played on classical 19×19 boards; many games also include unplayed analysis variations which we do not use in this work. These 9524 games contain 458,182 total comments, with a median length of 14 words.

2.2 Feature Extraction

We convert board states and comments into binary feature vectors using two methods: (1) *pattern-based* feature extraction, which checks for the ground truth presence of features from the board state, and (2) *keyword-based* feature extraction, which converts comments into bag-of-words representations based on domain-specific keywords.

Pattern-Based We define a set of rules to determine which game phenomena are present in a given board state, including: *cuts*, *eyes*, *ladders*, and *walls*. For example, we decide that a *wall* is present when four stones of the same color are placed in a row adjacent to one another. Because patterns like *wall* and *cut* are often imprecisely defined, these definitions may not align perfectly with player intuitions; we therefore provide additional details for each phenomena in the appendix. We do not attempt to write rule-based definitions of vaguer concepts like *sente* and *influence*.

Keyword-Based We scrape an online vocabulary of domain-specific terminology[2] and find the 30 most common terms in our natural language annotations. We then convert each comment into a 30-dimensional binary feature vector representing whether or not it contains these keywords; we additionally include features based on 60 control words, chosen according to frequency statistics, which are further subdivided into function and content words. Our wordlist and details about our selection of control words can be found in the appendix.

[1] https://gtl.xmp.net

[2] https://senseis.xmp.net/?GoTerms

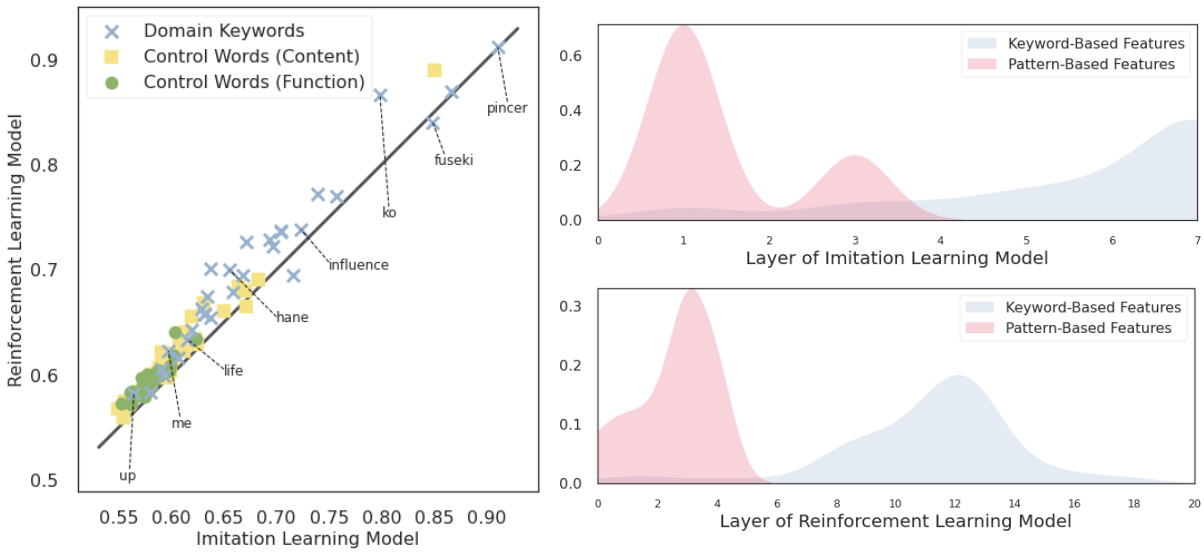


Figure 3: Results for the imitation learning and reinforcement learning agents are highly correlated. (Left) Scatterplot of ROC AUC values for linear probes trained to predict the presence of domain-specific keywords in move-by-move annotations. Keywords but not control words are predictable from the intermediate layers of both models. Words to the left of the solid line ($y = x$) are better predicted from the reinforcement learning model. (Right) Kernel density estimates showing where information is best represented in the policy networks (cf. §3.3). For both policy networks, pattern-based features are encoded in early layers, while keyword-based features are most easily extracted from later layers. Layer 0 denotes the input board representation for both models.

Having both pattern-based and keyword-based features captures a trade-off between precision and coverage. Writing rules for pattern-based features is labor-intensive and essentially impossible for many game concepts. Meanwhile, keyword-based features are inherently noisy: comments often mention phenomena which didn't actually occur in the game, and common structures like *atari* and *eyes* are frequently left unmentioned because the annotator and players already know they exist. Nonetheless, we find that probes are capable of predicting the presence of domain-specific keywords with significantly better-than-chance accuracy.

3 Methods

3.1 Policy Networks

We analyze two agents: (1) an imitation learning agent using the architecture described in Clark and Storkey (2015), and (2) a pre-trained ELF OpenGo model (Tian et al., 2017, 2019), which is an open-source, reinforcement learning agent similar to AlphaGo Zero (Silver et al., 2017). Our imitation learning model was trained on 228,000 games and achieved a rating of 1K ($\approx$ 1900 ELO) on the Online Go Server (OGS),[3] where it played against a combination of humans and computers until its rating stabilized. ELF OpenGo reports a self-play ELO over 5000, but this metric is inflated (Tian et al., 2019). Although we refer to these agents by their training procedure (i.e., imitation vs. reinforcement), there are several other differences between the models. One possible source of variance between agents involves the format of the board state representation. Following Clark and Storkey (2015), our imitation learning model takes as input a $19 \times 19 \times 7$ binary matrix. Of the seven planes, six represent the positions of stones, divided by color and the number of *liberties*; the seventh plane represents *ko* information. Meanwhile, the reinforcement learning model's $19 \times 19 \times 17$ input contains a partial history of the game state.

3.2 Linear Probes

Given a board state and paired feature vector as described in Section 2.2, we compute intermediate representations by feeding the board state into frozen policy networks. To predict each feature of interest, we run logistic regression independently on each layer of each policy network, including the raw board state. In other words, for each policy network, we train $F \times L \times k$ classifiers, where F is the number of features, L is the number of layers, and k is the parameter for k-fold cross-validation, as discussed in the following section.

[3] https://online-go.com

Domain Word	Imitation	Reinforcement	Rough Definition
Pincer	0.91	0.91	attack on a corner approach
Joseki	0.87	0.87	fixed local sequences of moves
Fuseki	0.85	0.84	opening
Ko	0.80	0.86	repetitive capture sequence
Wall	0.70	0.74	sequence of stones in a row
Atari	0.69	0.73	threat to capture
Eye	0.67	0.73	surrounded empty space
Cut	0.64	0.65	block two groups from connecting
Me	0.60	0.62	another word for *eye*
Down	0.60	0.60	toward the edge of the board
Point	0.59	0.61	specific locations on the board; or, the score
Force	0.58	0.58	requiring immediate response
Up	0.56	0.58	toward the center of the board

Table 1: ROC AUC values for a subset of domain words in both the imitation learning and reinforcement learning models. Higher values correspond to more predictable words. Domain words with the highest values represent relatively straightforward corner patterns (*pincer*), while keywords with the lowest values (*force*, *up*) are polysemous with commonly used non-domain-specific meanings. See Table 2 in the appendix for additional ROC AUC values.

3.3 Metrics

We seek to answer two questions: (1) *what* information is represented in the policy networks, and (2) *where* is this information represented? To answer the first question, we compute the area under the receiver operating characteristic curve (ROC AUC) for each linear probe. Specifically, for each layer, we compute the average ROC AUC after 10-fold cross-validation and then take the maximum average value across layers. Features with high ROC AUC are said to be *represented* by a model, because they are linearly extractible from some intermediate layer of its policy network. To answer the second question, we compute the layer at which each feature has its highest ROC AUC value; we then apply 10-fold cross-validation, summarize the counts for each feature in a histogram, and compute a kernel density estimate (KDE) for visualization.

4 Results

We find that domain-specific keywords are significantly more predictable than control words, with $p = 1.8 \times 10^{-5}$ under the Wilcoxon signed-rank test. As shown in Figure 3 (Left) and Table 1, the keyword with the highest ROC AUC value across both models is *pincer*, which denotes a relatively straightforward corner pattern. Meanwhile, low-valued domain words like *me* and *up* are polysemous with non-domain-specific meanings and therefore difficult to predict. While content and function control words have roughly similar distributions, some content words are noticeably more predictable; for example, *opponents* is the highest-valued control word with ROC AUC values of $(0.85, 0.89)$ as seen in Figure 3 (Left). Such control words are likely predictable due to correlations with certain domain-specific concepts.

ROC AUC values for the two models are strongly correlated, with Pearson's coefficient $\rho = 0.97$. Figure 3 (Left) shows that for most keywords, the reinforcement learning model slightly outperforms the imitation learning model. Furthermore, keywords are significantly more predictable from the imitation learning model than from a randomly initialized baseline with identical architecture ($p = 5.6 \times 10^{-16}$). Some words like *ko* are noticeably more predictable from the reinforcement learning model, possibly due to differences in input board state representations (cf. §3.1); further discussion of this point can be found in the appendix.

Consistent with our knowledge that pattern-based features can be obtained by applying simple rules to the raw board state, we find that pattern-based features are encoded in early layers of both models, as shown in Figure 3 (Right). Meanwhile, keyword-based features are most easily extracted from later layers, suggesting that they correlate with high-level abstractions in the policy network. Generally, pattern-based features are much more predictable than keyword-based features, with average ROC AUC values of $(0.96, 0.98)$ and

$(0.68, 0.70)$, respectively. As discussed in Section 2.2, this discrepancy can largely be attributed to the noisiness inherent in natural language data.

5 Related Work

Jhamtani et al. (2018) propose a similarly-sized dataset of move-by-move chess commentary. Rather than using this commentary for model interpretability, though, Jhamtani et al. (2018) attempt to predict whole comments from raw board states. Zang et al. (2019) use the same dataset to jointly train a policy network and language generation model with a shared neural encoder, but again focus on the pedagogical application of commentary generation rather than interpretation of the policy network. Similar work has focused on generating sportscasts in the Robocup domain (Chen and Mooney, 2008; Liang et al., 2009; Mei et al., 2016).

Our primary methodology is linear probing (Ettinger et al., 2016; Manning et al., 2020), which has commonly been used to study the intermediate representations of language models like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019). One classic result in this area shows that early layers of contextual language models correlate best with lexical-syntactic information such as part of speech, while later layers correlate with semantic information like proto-roles and coreference (Tenney et al., 2019). Recent work on control tasks (Hewitt and Liang, 2019), minimum description length (Voita and Titov, 2020), and Pareto probing (Pimentel et al., 2020) has focused on improving the methodological rigor of this paradigm. Although linear probing is fundamentally a correlational method, other recent work has focused on whether information which is easily extractable from intermediate layers of a deep network is causally used during inference (Elazar et al., 2021; Lovering et al., 2021).

Most related to our work are contemporary studies by McGrath et al. (2021) and Forde et al. (2022), which apply probing techniques to the games of chess and Hex, respectively. McGrath et al. (2021) use linear probes to predict a large number of pattern-based features throughout the training of an AlphaZero agent for chess. Meanwhile, Forde et al. (2022) train linear probes for pattern-based features on an AlphaZero agent for Hex and run behavioral tests to measure whether the agent "understands" these concepts. Comparatively, our work uses fewer features than McGrath et al. (2021) and does not make causal claims about how representations are used during inference, as in Forde et al. (2022); however, to the best of our knowledge, our work is the first of its kind to use features derived from natural language in conjunction with probing techniques for policy interpretability.

6 Conclusion

We presented a new dataset of move-by-move annotations for the game of Go and showed how it can be used to interpret game-playing agents via linear probes. We observed large differences in the predictability of pattern-based features, which are extracted from the board state, and keyword-based features, which are extracted from comments. In particular, pattern-based features were easily extracted from lower layers of the policy networks we studied, while keyword-based features were most predictable from later layers. At a high level, this finding reinforces the intuition that written annotations describe high-level, abstract patterns that cannot easily be described by a rule-based approach. Accordingly, we argue there is much to learn from this annotation data: future work might attempt to correlate policy network representations with richer representations of language, such as those provided by a large language model. Future work might also explore whether similar approaches could be used to improve game-playing agents, either by exposing their weaknesses or providing an auxiliary training signal. We also expect similar approaches may be viable in other reinforcement learning domains with existing natural language data.

Acknowledgements

We are grateful to Kevin Yang for his work on the imitation learning model, and to Rodolfo Corona and Ruiqi Zhong for their contributions to an early version of this project. We thank Roma Patel, the members of the Berkeley NLP Group, and our anonymous reviewers for helpful suggestions and feedback. This work was supported by the DARPA XAI and LwLL programs and a National Science Foundation Graduate Research Fellowship.

References

David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135.

Christopher Clark and Amos Storkey. 2015. Training deep convolutional neural networks to play Go. In

International conference on machine learning, pages 1766–1774. PMLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2021. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175.

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. In *Proceedings of the 1st workshop on evaluating vector-space representations for nlp*, pages 134–139.

Jessica Zosa Forde, Charles Lovering, George Konidaris, Ellie Pavlick, and Michael L. Littman. 2022. Where, when which concepts does AlphaZero learn? lessons from the game of Hex. In *AAAI Workshop on Reinforcement Learning in Games*.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, Graham Neubig, and Taylor Berg-Kirkpatrick. 2018. Learning to generate move-by-move commentary for chess games from large-scale social forum data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1661–1671.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99.

Charles Lovering, Rohan Jha, Tal Linzen, and Ellie Pavlick. 2021. Predicting inductive biases of pretrained models. In *International Conference on Learning Representations*.

Christopher D Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054.

Thomas McGrath, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. 2021. Acquisition of chess knowledge in alphazero. *arXiv preprint arXiv:2111.09259*.

Hongyuan Mei, TTI UChicago, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of NAACL-HLT*, pages 720–730.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Tiago Pimentel, Naomi Saphra, Adina Williams, and Ryan Cotterell. 2020. Pareto probing: Trading off accuracy for complexity. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3138–3153. Association for Computational Linguistics.

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

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601.

Yuandong Tian, Qucheng Gong, Wenling Shang, Yuxin Wu, and C. Lawrence Zitnick. 2017. Elf: An extensive, lightweight and flexible research platform for real-time strategy games. In *Advances in Neural Information Processing Systems*, pages 2656–2666.

Yuandong Tian, Jerry Ma, Qucheng Gong, Shubho Sengupta, Zhuoyuan Chen, James Pinkerton, and Larry Zitnick. 2019. ELF OpenGo: An analysis and open reimplementation of alphazero. In *International Conference on Machine Learning*, pages 6244–6253. PMLR.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196.

Hongyu Zang, Zhiwei Yu, and Xiaojun Wan. 2019. Automated chess commentator powered by neural chess engine. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5952–5961.

A Dataset Statistics

The most common domain-specific terms appear in more than 15K comments, as shown in Figure 4.

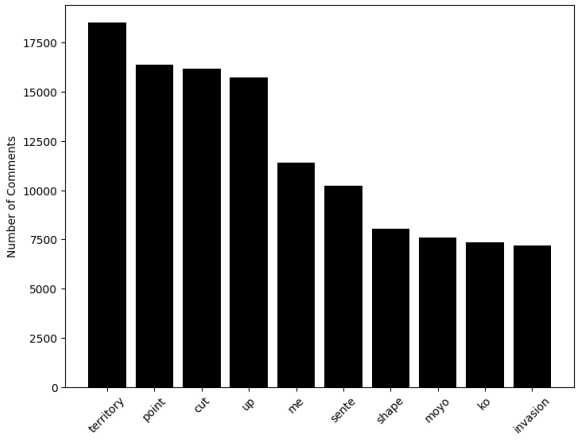


Figure 4: Histogram showing the number of comments that contain each of the ten most common domain-specific words.

B Pattern-Based Feature Extraction

As described in Section 2.2, we extract features from the board state using a set of hand-crafted rules. These rules may not align perfectly with player intuitions, which can be hard to formulate concisely, and so are presented here for full detail.

Cut We define *cuts* as moves that prevent the opponent from connecting two disconnected groups on their next move. To avoid labelling squares where a play would be immediately capturable, we also require that a cut have at least two liberties. Note that this definition permits non-diagonal cuts.

Eye We define *eyes* as connected groups of empty squares that are completely surrounded by stones of the same color. We require there be no enemy stones in the same surrounded region, so this definition fails to capture eyes that surround *dead* stones.

Ladder The term *ladder* describes the formation shown in Figure 2c. Since human players can usually predict who wins a ladder, they rarely play out the capturing race. For this reason, we do not look for ladder formations, but instead label moves that would start or continue a ladder. Specifically, we label a square for the ladder feature if it is the singular liberty of a friendly group of stones and a play at the square results in the group having exactly two liberties. We do not count trivial ladders that lie at the edge of the board.

Wall We define a *wall* as a connected row or column with four or more stones of the same color.

C Keywords and Control Words

Keywords We choose the first thirty most frequent terms (cf. Table 1) from our vocabulary of domain-specific terminology as keywords: *territory, point, cut, sente, up, me, moyo, shape, ko, invasion, influence, wall, joseki, eye, alive, gote, life, pincer, aji, thickness, base, atari, connected, hane, tenuki, down, overplay, force, reading, fuseki*.

Control Words Our control words consist of the thirty most frequent words in our dataset, as well as thirty words uniformly distributed according to the same frequency as the keywords: *the, is, to, this, white, black, and, you, at, for, in, move, it, of, but, not, be, have, play, that, on, good, here, if, better, can, would, now, should, stones, looking, wanted, opponents, wasnt, defending, save, youre, answer, three, fine, feel, place, lose, bit, possibility, attacking, likely, leaves, shouldnt, question, lost, threat, almost, theres, continue, trying, hope, just, exchange, before*. We further subdivide the control words based on whether or not they appear in the NLTK stopword list,[4] which we use as a rough proxy for distinguishing between function and content words.

D Additional Results

We additionally report de-aggregated ROC AUC values for each keyword across layers, as shown in Figures 5-7. These figures show the raw data used to compute the kernel density estimates in Figure 3, which show that natural language features are most easily extracted from later layers of both models. We note in Figure 5 that *ladders* are the most difficult pattern-based feature to predict, which is consistent with our knowledge that many Go-playing agents fail to correctly handle ladders without special feature engineering (Tian et al., 2019). Anecdotally, our imitation learning model often failed to play ladders correctly; this is consistent with the finding that ladders are more predictable from the reinforcement learning model. Future work might investigate whether this probing framework could be used to effectively predict model behavior in situations like these, as in Forde et al. (2022) for the game of Hex.

[4] https://gist.github.com/sebleier/554280

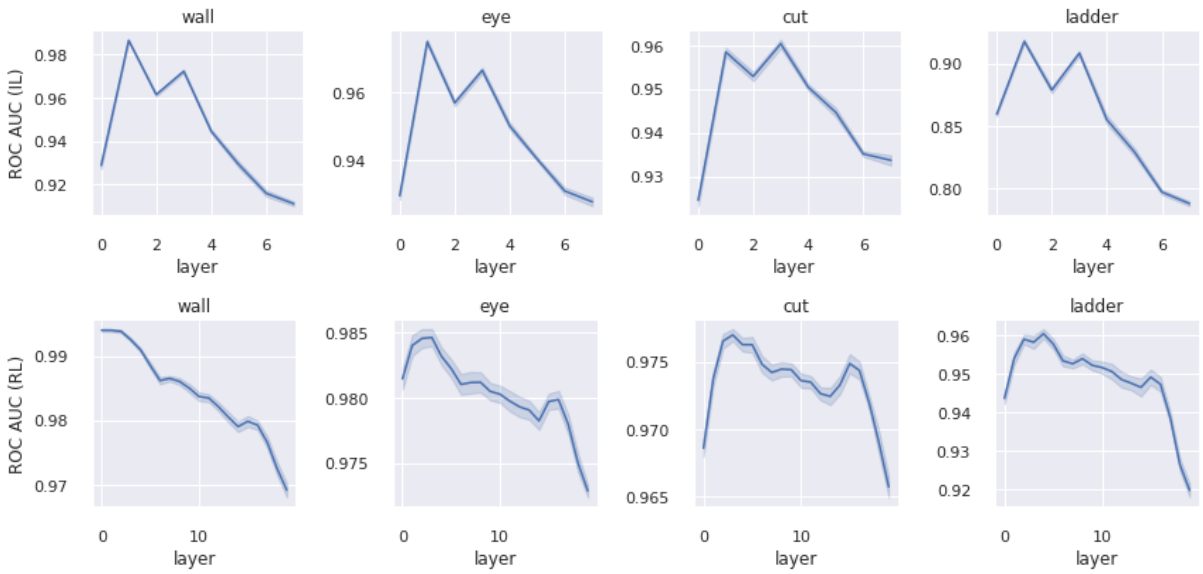


Figure 5: Plots of ROC AUC values for pattern-based features in the imitation learning model (top) and the reinforcment learning model (bottom). Among the four pattern-based features we consider, *ladders* have by far the lowest ROC AUC values. As noted in Tian et al. (2019), *ladders* are a known challenge for Go agents, requiring special feature engineering in Silver et al. (2016). Therefore, perhaps it is unsurprising that *ladders* were the most difficult pattern-based feature to predict.

E Major Differences Between Imitation and Reinforcement Learning Models

While most keywords have similar ROC AUC values across models, *ko*, *eye*, *atari*, and *overplay* have a noticeably higher ROC AUC values under the reinforcement learning model (cf. Table 1). However, this discrepancy is not obviously attributable to the difference in training procedures (i.e., imitation vs. reinforcement). As described in Section 3.1, the two models use different input state representations, which differ in their encoding of *ko* and *liberty* information, which is used to determine whether *eyes* and *atari* exist. Such architectural differences may explain discrepancies across models, but do not account for words like *overplay*; playing strength is another possible (but not confirmed) source of these discrepancies.

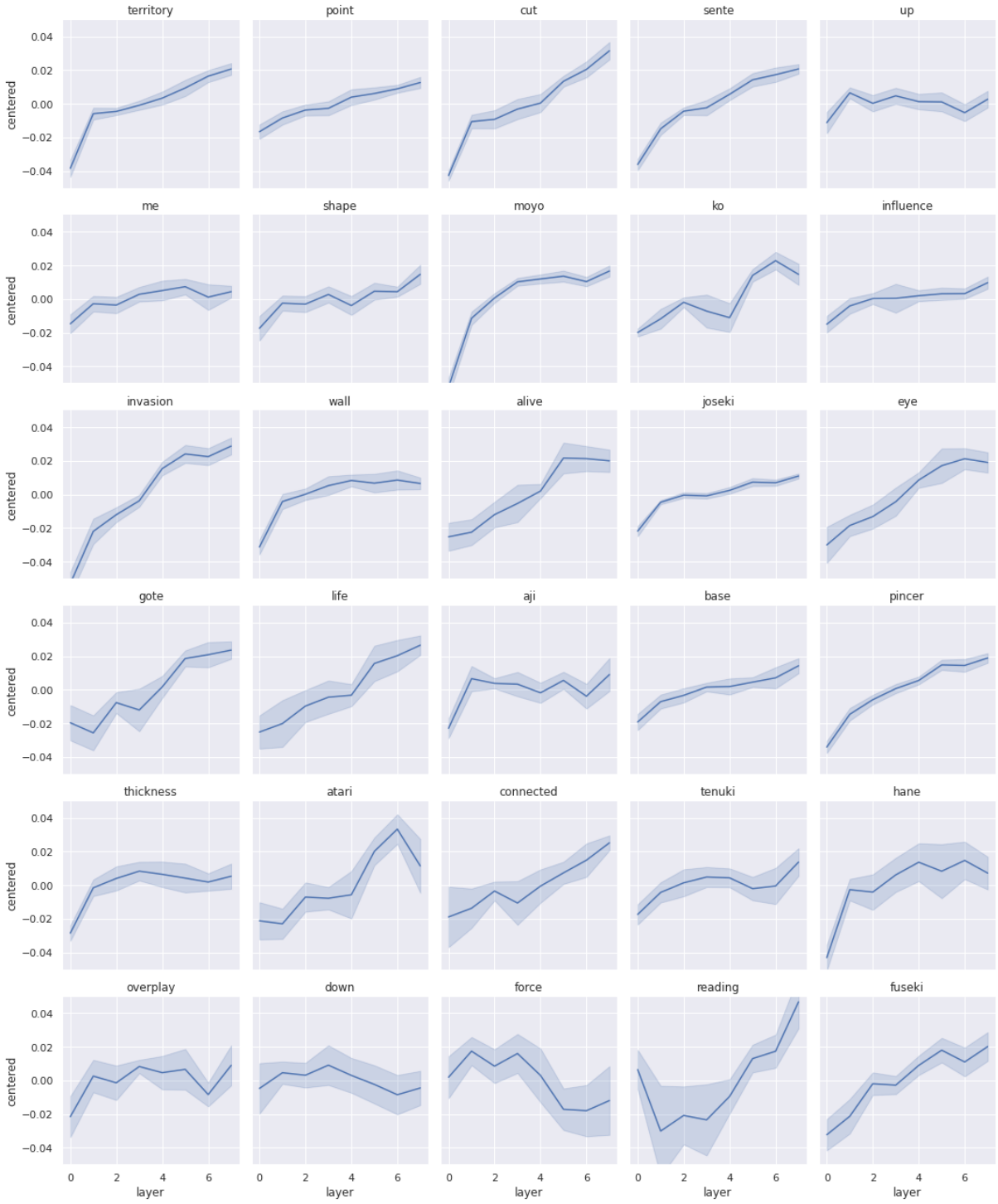


Figure 6: Plots of centered ROC AUC values for keywords in the imitation learning model. For most keywords, linear probe performance peaks at mid-to-late layers of the imitation learning model.

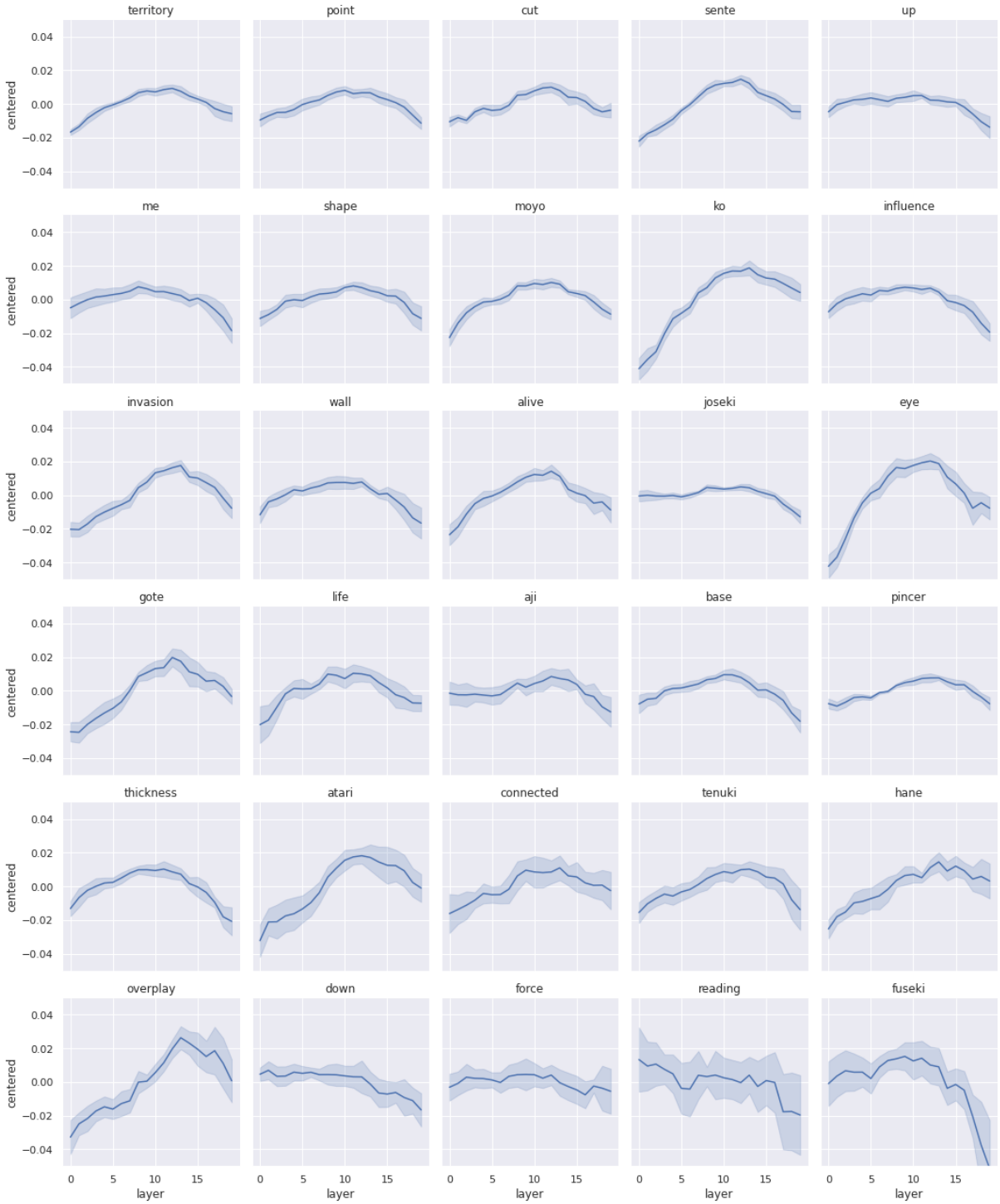


Figure 7: Plots of centered ROC AUC values for keywords in the reinforcement learning (RL) model. While noisier than the imitation learning model, probe performance still tends to peak at mid-to-late layers of the RL model and declines at the final layers. Figure 3 aggregates these results alongside pattern-based feature classifiers.

Domain Word	Imitation	Reinforcement	Rough Definition
Pincer	0.91	0.91	attack on a corner approach
Joseki	0.87	0.87	fixed local sequences of moves
Fuseki	0.85	0.84	opening
Ko	0.80	0.86	repetitive capture sequence
Base	0.76	0.77	starter eye space
Moyo	0.74	0.77	sphere of influence
Influence	0.72	0.74	long-range effect of stones
Reading	0.72	0.70	calculating an upcoming sequence
Wall	0.70	0.74	sequence of stones in a row
Thickness	0.70	0.74	strength of a group of stones
Invasion	0.70	0.72	attack on enemy territory
Atari	0.69	0.73	threat to capture
Eye	0.67	0.73	surrounded empty space
Gote	0.67	0.69	loss of initiative
Tenuki	0.66	0.68	non-local response
Hane	0.66	0.70	move that "reaches around" or bends
Overplay	0.64	0.70	overly aggressive move
Cut	0.64	0.65	block two groups from connecting
Alive	0.63	0.67	cannot be captured
Territory	0.63	0.66	controlled empty space
Aji	0.63	0.66	possibilities left in a position
Sente	0.63	0.66	initiative
Shape	0.62	0.64	quality of a group of stones
Life	0.62	0.63	inability to be captured
Connected	0.61	0.62	adjacent or nearby stones
Me	0.60	0.62	another word for *eye*
Down	0.60	0.60	toward the edge of the board
Point	0.59	0.61	specific locations on the board; or, the score
Force	0.58	0.58	requiring immediate response
Up	0.56	0.58	toward the center of the board

Table 2: ROC AUC values for all domain words in both the imitation learning and reinforcement learning models. Domain words with the highest values represent relatively straightforward corner patterns (*pincer*), while keywords with the lowest values (*force*, *up*) are polysemous with commonly used non-domain-specific meanings.