Comparing Optimization Targets for Contrast-Consistent Search

Hugo Fry
Independent
hugo.fry@fryfamily.co.uk

Seamus Fallows
Independent
seamusfallows1@gmail.com

Ian Fan
Independent

Jamie Wright
Oxford University

Nandi Schoots
King's College London

Abstract

We investigate the optimization target of Contrast-Consistent Search (CCS), which aims to recover the internal representations of truth of a large language model. We present a new loss function that we call the Midpoint-Displacement (MD) loss function. We demonstrate that for a certain hyper-parameter value this MD loss function leads to a prober with very similar weights to CCS. We further show that this hyper-parameter is not optimal and that with a better hyper-parameter the MD loss function tentatively attains a higher test accuracy than CCS.

1 Introduction

The increased deployment of large language models to real-world applications has continued to highlight the discrepancy between the growing capabilities of language models and our limited understanding of their behavior [OpenAI, 2023].

It is widely believed that language models have internal representations encoding knowledge of the world [Li et al., 2023]. In order to deploy such systems safely, a better understanding of these internal representations is needed. In particular, language models will often confidently output false statements or 'hallucinate' in a number of settings [Ji et al., 2023]. It is therefore important to develop techniques aimed at discovering internal representations of truth.

Mechanistic interpretability aims to reverse engineer the algorithms that the model weights implement at the level of individual neurons [Cammarata et al., 2020, Wang et al., 2022]. Other interpretability work more broadly aims to develop automatic techniques for probing and modifying human-interpretable concepts from a model's activations [Ghorbani et al., 2019, Wang et al., 2022].

Recent work in the field of interpretability has presented contrast-consistent search (CCS) [Burns et al., 2022] as an unsupervised method for extracting knowledge from the hidden states of large language models. It is able to learn the truth value of statements using the negation consistency property of truth: a statement and its negation must have opposite truth values. By comparing the model representations corresponding to these two statements, one can find a direction in activation space that satisfies this consistency constraint.

In this paper, we investigate the optimization target of CCS. Our contributions are as follows:

- In Section 3.1, we present some conceptual clarifications on how CCS classifies truth and how it should be interpreted.
- In Section 3.2, we present a heuristic explanation of CCS's optimization target. Motivated by this explanation, we introduce the Midpoint-Displacement (MD) loss function as a proxy optimisation target for CCS.

Socially Responsible Language Modelling Research (SoLaR) NeurIPS 2023

- In Section 4, we present an experimental comparison of many loss functions across a number of datasets and models. In particular, we demonstrate that, for a certain choice of the hyper-parameter, the truth direction found by the MD loss function is a good proxy for CCS, as measured by cosine similarity.
- Further, we tentatively show that this hyper-parameter is not optimal and that with a better hyper-parameter the MD loss function outperforms the current state of the art (CCS), along with all other loss functions that we have tested.

2 Background

Contrast-Consistent Search (CCS), developed by Burns et al. [2022], is an unsupervised method for extracting knowledge from the hidden states of large language models. Given only unlabelled model activations, CCS is able to accurately classify statements according to their truth value. It does this by utilising the negation consistency property of truth: a statement and its negation must have opposite truth values. Framed in terms of probabilities, given the probability p that a proposition is true, the probability its negation is true is $1 - p$. CCS works by finding a direction in activation space that satisfies this consistency constraint.

Dataset. We take a dataset of *contrast pairs* $\{(x_i^+, x_i^-)\}_{i=1}^n$ consisting of natural language statements x_i^+ and their logical opposites x_i^-. The pairs are formed by taking a question q_i and appending with one of two mutually exclusive answers. Contrast pairs are fed to a pre-trained language model to obtain a set of representations $\{(\phi_i^+, \phi_i^-)\}_{i=1}^n$, where $\phi_i^\pm := \phi(x_i^\pm) \in \mathbb{R}^d$ is the activation vector of a particular layer for the input $x_i^\pm$. These representations are then used to train a linear classifier according to an objective designed to enforce negation consistency on contrast pairs.

Below is an example of a contrast pair:

q_i: *"Are cats mammals?"*
x_i^-: *"Are cats mammals? No."*
x_i^+: *"Are cats mammals? Yes."*

In order to avoid training the classifier to simply detect the presence of the mutually exclusive answers, the sets $\{\phi_i^+\}$ and $\{\phi_i^-\}$ should be independently normalized. The normalized representations are given by

$$\tilde{\phi}_i^\pm := \frac{\phi_i^\pm - \mu^\pm}{\sigma^\pm}, \tag{1}$$

where $(\mu^\pm, \sigma^\pm)$ are the means and standard deviations of the respective sets. For convenience, we will omit the tilde and simply use $\phi_i^\pm$ to represent the normalized representations.

Loss Function. A linear classifier $p_{\theta,b} : \phi \to \sigma(\theta^T \phi + b)$ is trained on the normalized activations, where σ is the sigmoid function, θ is a vector of weights and b is a bias. The loss function is given by

$$L_{\text{CCS}}(\theta, b) := \frac{1}{n} \sum_{i=1}^{n} \left[1 - p_{\theta,b}(\phi_i^+) - p_{\theta,b}(\phi_i^-)\right]^2 + \min\left\{p_{\theta,b}(\phi_i^+), p_{\theta,b}(\phi_i^-)\right\}^2. \tag{2}$$

The first term encourages the classifier to find features that are negation-consistent and the second term is included to disincentivise the degenerate assignment $p_{\theta,b}(\phi_i^+) = p_{\theta,b}(\phi_i^-) = 0.5$.

Metrics. The accuracy of CCS is calculated using the ground-truth dataset labels. That is, CCS probers are trained in an unsupervised way, but evaluated using supervised labels.

Inference. To make a prediction on an example q_i, after training, the average

$$\bar{p}_i(q_i) := \frac{1}{2}(p_{\theta,b}(\phi_i^+) + (1 - p_{\theta,b}(\phi_i^-))) \tag{3}$$

is compared to 0.5 with $\bar{p}_i > 0.5$ corresponding to either the answer "yes" or "no" based on whichever gives the maximum accuracy on a given test set.[1]

Burns et al. [2022] show that a CCS prober trained with the above loss function outperforms zero-shot outputs of the model with a mean accuracy of 71.2%, against 67.2% for zero-shot.

[1]Note that this supervised method is used for simplicity of evaluation; Burns et al. [2022] describe a completely unsupervised method based on conjunctions for determining this assignment.

3 Methods: Introducing New Loss Functions

In this section we introduce an alternative loss function to CCS. First, we provide some conceptual clarification and address a possible misconception about how CCS works. The code used to run our experiments can be found at `https://github.com/ash-ai-safety-hub/g3-nandi`.

3.1 Clarifying CCS

A natural guess for how CCS is able to accurately classify truth is that the normalized model representations are approximately clustered according to truth and that CCS is able to find a hyperplane that separates these two clusters. However, this explanation turns out to be incorrect. In Appendix A.1 we provide a specific example showing that model activations do not in fact cluster in this way. Recall that after training, an example q_i is classified according to $\bar{p}(q_i) > 0.5$. Using equation 3, this condition reduces to

$$\begin{aligned} &\sigma(\theta^T \phi_i^+ + b) > \sigma(\theta^T \phi_i^- + b) \\ \Rightarrow\ &\theta^T(\phi_i^+ - \phi_i^-) > 0. \end{aligned}$$

We see that CCS is classifying only according to the displacement vectors $(\phi_i^+ - \phi_i^-)$. Since these are translation invariant, CCS does not require the contrast pairs to be separable by a hyperplane.

Additionally, the original paper presents CCS as learning probabilities for the truth values of contrast pairs. We suggest abandoning this probabilities framing. We show in Appendix A.2 that CCS can still perform well even when the probabilities are strongly clustered around 0.5. This paints a different picture to that presented in [Burns et al., 2022].

3.2 Midpoint-Displacement (MD) Loss Function

In this section we present a heuristic explanation of CCS's optimization target and use this explanation to introduce a new loss function. Note first that the CCS loss function incentivises increasing the separation of the prober outputs of contrast pairs $|p(\phi_i^+) - p(\phi_i^-)|$. Consider a CCS prober $p(\phi) = \sigma(\theta^T \phi + b)$ in which θ is constrained to a fixed norm[2] $|\theta| = c$.

Using the normalized weight vector $\hat{\theta}$, we define the following quantities:

$$\begin{aligned} u_i &:= \phi_i^+ - \phi_i^-, \text{ and} \\ \sigma_d^2 &:= \frac{1}{n} \sum_i (\hat{\theta}^T u_i)^2. \end{aligned} \tag{4}$$

Here u_i is the displacement between the activations of a contrast pair and σ_d^2 is the mean square separation of the activations of the contrast pairs along the direction θ.

Furthermore, we analogously define

$$\begin{aligned} v_i &:= \phi_i^+ + \phi_i^-, \text{ and} \\ \sigma_m^2 &:= \frac{1}{n} \sum_i (\hat{\theta}^T v_i)^2. \end{aligned} \tag{5}$$

Here $\dfrac{v_i}{2}$ is the midpoint of the activations of a contrast pair and $\dfrac{\sigma_m^2}{4}$ is the mean square value of the midpoint of the activations of the contrast pairs along the direction θ.

In order for CCS to increase the difference between prober outputs, one might expect that CCS finds a direction that increases the difference of the prober inputs (since sigmoid is a monotonically increasing function). That is to say, one might expect CCS will find a direction that maximises σ_d^2.

However, if σ_m^2 is much larger than σ_d^2, then the input to the sigmoid for each contrast pair (i.e. $\theta^T \phi_i^+ + b$ and $\theta^T \phi_i^- + b$) will be pushed into the same saturation regime of the sigmoid. This results in a lower difference in prober outputs of contrast pairs, which in turn results in a trade off between maximising σ_d^2 while minimising σ_m^2.

[2]We use $|...|$ to denote the Euclidean norm throughout this paper.

It should be stressed that this trade-off between σ_d^2 and σ_m^2 is purely an artifact of the double saturation of sigmoid used in the CCS prober. Since this trade off should occur no matter what $|\theta| = c$ is constrained to, we propose that the unconstrained CCS prober is in general optimising for some balance between σ_d^2 and σ_m^2.

To test this hypothesis, we propose a new loss function and demonstrate that this new loss function is a good proxy optimisation target for CCS. The new loss function is given by

$$L_{\text{MD}} = (\lambda - 1)\sigma_d^2 + \lambda \cdot \sigma_m^2, \tag{6}$$

where $\lambda \in [0, 1]$ is a hyper-parameter controlling the relative trade off between σ_d^2 and σ_m^2, and the weight vector θ is constrained to satisfy $|\theta| = 1$.

In Appendix B we introduce two new loss functions: the Mean Absolute (MA) loss function and the Square Mean Root (SMR) loss function. In the following experiments we compare the MD loss function with a number of other loss functions, including the CCS, MA and SMR loss functions along with Principal Component Analysis (PCA).

4 Results: Comparison of MD with CCS

In this section we investigate the empirical similarity between the MD and CCS loss functions. The MD-CCS and MD-Accuracy (MD-Acc) loss functions are both trained using the Midpoint-Displacement loss function but with different hyper-parameter searches. The implementation details of our experiments can be found in Appendix C.

For comparison, we include probers trained using a variety of loss functions. The MA and SMR loss functions are both based on taking the mean displacements (further details can be found in Appendix B). The PCA loss function identifies the first principal component of the displacements. The random probers (Rand.) are found by randomly initializing 10 weight vectors and taking the average resulting accuracy. Lastly, the supervised probers (Superv.) are probers trained on labelled data with the same structure as the CCS probers, $p(\phi) = \sigma(\theta^T \phi + b)$.

In Table 1 we show the test accuracies of probers trained using the MD loss function on various datasets and models. We tentatively find that the accuracies of the new probers are similar to those achieved by CCS, and often out-perform CCS. Note that MD-Acc probers get a higher test accuracy than both the MD-CCS and CCS probers for three out of four models, for an average difference of around 4%.

Model	Loss Function							
	CCS	MD-CCS	MD-Acc	MA	SMR	PCA	Rand.	Superv.
UQA (E)	0.6863	0.6902	0.7414	0.7399	**0.7419**	0.7383	0.6363	0.8839
UQA (D)	**0.8305**	0.8200	0.8180	0.7550	0.7460	0.7525	0.6286	0.9140
DeBERTa	0.7740	0.7855	**0.8735**	0.8650	0.8585	0.8605	0.7288	0.9135
GPT-Neo	0.5510	0.5755	**0.5898**	0.5820	0.5555	0.5737	0.5603	0.7580
Average	0.7105	0.7178	**0.7557**	0.7355	0.7255	0.7313	0.6385	0.8674

Table 1: We compare test accuracies of different loss functions averaged over five datasets, using the activations of a number of models. For each row we have emboldened the loss function that obtained the highest average test accuracy, not including the supervised loss. The (E) and (D) labels refer to the encoder and decoder layers of the UQA model.

In Table 2 we find that the average cosine similarity between the weight vector of the CCS prober and the weight vector of the prober trained using our new MD method is about 0.63. For reference, the probability of two uniformly sampled 1024-dimensional unit vectors having a cosine similarity of 0.63 or higher is approximately 10^{-237}. Note that the CCS probers had an average cosine similarity with themselves of only 0.78. This suggests that the MD-CCS loss function is a good proxy optimization target for CCS.

The only difference between the MD-CCS and MD-Acc loss functions is the value of their hyper-parameter λ that controls the relative trade off between σ_d^2 and σ_m^2. Since they give very different cosine similarities, we find the similarity of MD to CCS is dependent on this trade off.

Model	Loss Function							
	CCS	MD-CCS	MD-Acc	MA	SMR	PCA	Rand.	Superv.
UQA (E)	0.8359	**0.7034**	0.2995	0.1448	0.1991	0.2406	0.0222	0.2583
UQA (D)	0.8787	**0.7269**	0.5303	0.1687	0.2432	0.1792	0.0228	0.6014
DeBERTa	0.8643	**0.6209**	0.2786	0.2309	0.2024	0.0741	0.0202	0.4617
GPT-Neo	0.5277	**0.4830**	0.4164	0.0226	0.0485	0.1901	0.0245	0.1347
Average	0.7767	**0.6336**	0.3812	0.1418	0.1733	0.1710	0.0224	0.3640

Table 2: We compute the average cosine similarities of the directions found using different loss functions to the directions of 20 CCS probers. We average over five datasets using the activations of four different models. For each row we have emboldened the loss function that obtained the highest average cosine similarity with CCS, not including the CCS loss function. Note that the (E) and (D) refer to the encoder and decoder layers of the UQA model.

In Appendix D, we include a breakdown of the experimental results for each dataset and model along with the hyper-parameters that are used for each loss function. It should be noted that the hyper-parameter used for the MD-CCS loss function does not change substantially between datasets or models. This suggests that the form of the proxy optimization target we have identified for CCS is robust to changes in the dataset and model.

5 Discussion

Our Midpoint-Displacement loss function sheds light on which aspects of the data CCS is picking up on. We find that the Midpoint-Displacement loss function produces probers that behave very similarly to those produced by CCS, as measured by cosine similarity. We verified that other reasonable loss functions, with comparable accuracy to CCS, do not get high cosine similarity to CCS.

Our findings suggest that the specific loss function formulation is not essential for performance, in that there are multiple loss functions that achieve high accuracy. Instead, the unique training data that CCS uses, which allows the identification of displacement between ϕ^+ and ϕ^-, is what drives the success of CCS.

5.1 Limitations

While the new loss functions we have proposed are unsupervised, they contain a hyper-parameter. The hyper-parameter is chosen using a supervised grid search and therefore supervised labels are currently required to use our loss functions. Additionally, the results obtained in this paper could be strengthened by testing across more datasets and models.

5.2 Future Work

Future work could also consider developing an unsupervised method of determining the hyper-parameter based on data statistics. This would eliminate the need for any supervised labels. Alternatively, we could establish a hyper-parameter with high transferability between datasets. Moreover, we propose that a small number of supervised examples may be sufficient to find a dataset-specific hyper-parameter.

In this paper we have evaluated similarity between probers through the metric of cosine similarity. Future work could additionally consider comparing the behavioral similarity between probers, by measuring their empirical pairwise agreement for contrast pairs in the test dataset.

5.3 Social Impact Statement

Our work is motivated by avoiding harmful behaviors of language models, such as deception. To detect deception in LLMs, we may compare model outputs to the model's latent knowledge. CCS is a nascent method for extracting truthfulness from latent representations. We aim to clarify CCS and we hope our results will inform future evaluation techniques that extract latent knowledge. We do not foresee harmful applications of our work.

Acknowledgments

This research was supported by the AI Safety Hub Labs programme.

References

Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. URL https://doi.org/10.5281/zenodo.5297715. If you use this software, please cite it using these metadata.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering Latent Knowledge in Language Models Without Supervision, December 2022.

Nick Cammarata, Shan Carter, Gabriel Goh, Chris Olah, Michael Petrov, and Ludwig Schubert. Thread: Circuits, mar 2020. ISSN 2476-0757. URL https://distill.pub/2020/circuits.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions, May 2019.

Amirata Ghorbani, James Wexler, James Zou, and Been Kim. arxiv.org/pdf/1902.03129.pdf, 2019.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention, 2021.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12):248:1–248:38, 2023. doi: 10.1145/3571730. URL https://doi.org/10.1145/3571730.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. Unifiedqa: Crossing format boundaries with a single qa system, 2020.

Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda B. Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=DeG07_TcZvT.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-1015.

Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, page 165–172, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450324090. doi: 10.1145/2507157.2507163. URL https://doi.org/10.1145/2507157.2507163.

OpenAI. Gpt-4 technical report, 2023.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems, February 2020.

Kevin Wang, Variengien Re, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the Wild: A Circuit for Indirect Object Identification in GPT-2 Small, 2022. URL https://arxiv.org/abs/2211.00593.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649–657, Cambridge, MA, USA, 2015. MIT Press.

A Clarifying CCS

A.1 CCS Does Not Determine Truth Using a Hyperplane

Let θ be the weight vector of the linear prober found by CCS and denote the unit-normalised weight vector by $\hat{\theta}$. We call this normalised weight vector 'the direction found by CCS'.

In Figure 1 we project the activations vectors ϕ_i corresponding to datapoints x_i onto the first principal component and onto the direction found by CCS (when trained on the decoder). We consider the activations for which CCS outputs 0.5 and show that this 'decision boundary' does not cleanly separate the true and false datapoints.

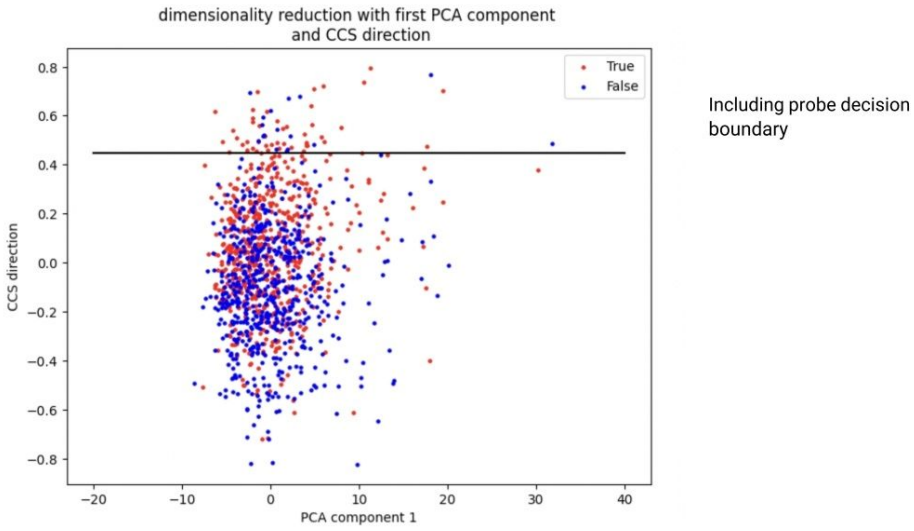
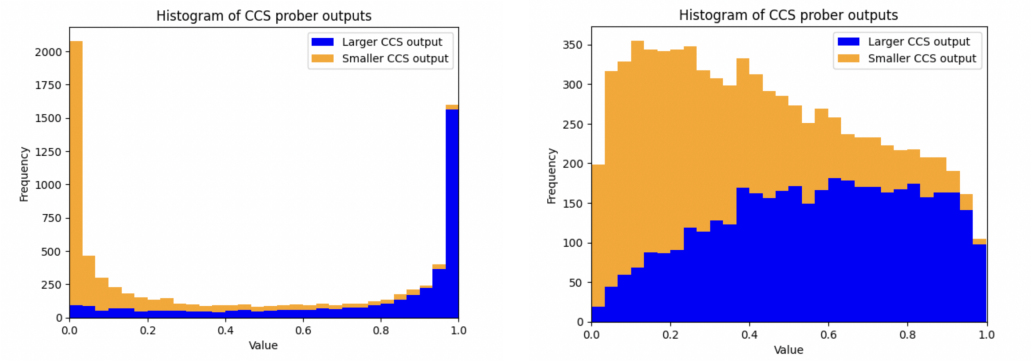


Figure 1: We consider activations of the T5-base model on the BoolQ train dataset. On the x-axis we plot the projections of the activations ϕ_i onto the first principle component and on the y-axis we plot the datapoints ϕ_i projected onto $\hat{\theta}$. We colour the datapoints ϕ_i by the (ground-truth) truth-labels 'true' (red) and 'false' (blue) of the datapoints x_i. The horizontal line indicates the inputs for which CCS outputs 0.5.

A.2 CCS is Miscalibrated

In Figure 2, we show histograms of CCS prober outputs evaluated on the last hidden state of both the encoder and decoder of UQA for the BoolQ dataset (compare to Figure 1 of Burns et al. [2022]). Interpreting the output of CCS as a probability would suggest much higher confidence for the encoder than the decoder. However, from Appendix D, the test accuracy on the encoder (0.523) is significantly worse than on the decoder (0.978). This suggests that the ouput of CCS should not be interpreted as probabilities.



(a) (b)

Figure 2: (a) Histogram displaying the CCS prober outputs evaluated on the last hidden state of the encoder of UnifiedQA T5-Large for the BoolQ dataset. (b) Histogram displaying the CCS prober outputs evaluated on the last hidden state of the decoder of UnifiedQA T5-Large for the BoolQ dtatset. Despite the encoder having a higher confidence in the prober outputs, the encoder has a lower test accuracy (0.523) than the decoder (0.978).

B Mean-Based Loss Functions

We define the displacement vector between the activations of a contrast pair as follows.

$$u_i = \phi_i^+ - \phi_i^-.$$

We define the mean and standard deviation of the modulus of the displacements along a direction θ as follows:

$$\mu = \frac{1}{n}\sum_i |\theta^T u_i|,$$

$$\sigma = \sqrt{\frac{1}{n}\sum_i (|\theta^T u_i| - \mu)^2}.$$

We then define the Mean Absolute (MA) loss function as

$$L_{\text{MA}} = (1-\lambda)\mu + \lambda \cdot \sigma.$$

The Square Mean Root (SMR) is given by

$$\mu_{\text{SMR}} = \sqrt{\frac{1}{n}\sum_i (\theta^T u_i)^2}.$$

Analogously, we define the SMR loss function to be

$$L_{\text{SMR}} = (1-\lambda)\mu_{\text{SMR}} + \lambda \cdot \sigma.$$

In both loss functions, we constrain $|\theta| = 1$.

C Implementation Details

For the experiments presented in this paper, all probers were trained for 1000 epochs with a learning rate of 0.01. All of the experiments were run for the following five datasets:

- IMDB [Maas et al., 2011]
- Amazon [McAuley and Leskovec, 2013]
- BoolQ [Clark et al., 2019]
- RTE [Wang et al., 2020]

- AG News [Zhang et al., 2015]

and the following three models:

- UnifiedQA T5-Large (encoder-decoder architecture) [Khashabi et al., 2020]
- GPT-Neo (decoder-only architecture) [Black et al., 2021]
- DeBERTa (encoder-only architecture) [He et al., 2021]

We use the hidden states of the encoder and decoder where available. In our experiments, we used 1000 examples from each dataset consisting of a 600-400 train-test split. For each loss function on each dataset and model, we use a grid search to determine the hyper-parameter that maximised train accuracy as follows:

1. Create eleven hyper-parameter values by splitting the interval $[0, 0.99]$ into eleven evenly spaced points (separated by 0.099).
2. For each of these values, we train three probers with a random seed. We evaluate the average train accuracy over the three probers.
3. We identify the hyper-parameter with the highest average train accuracy. Let this hyper-parameter be $\lambda^\star$.
4. Consider the new interval given by $[\lambda^\star - 0.099, \lambda^\star + 0.099] \cap [0, 0.99]$.
5. Repeat the process again on the new interval by splitting it evenly into eleven points.
6. Identify the hyper-parameter with the highest average train accuracy for the points on this new interval. This should calculate the hyper-parameter with a precision of roughly ± 0.02.

An analogous grid search is performed to identify the hyper-parameter used for the MD-CCS loss function, by finding the hyper-parameter that maximises average cosine similarity with 20 CCS probers. In the case of the MD-CCS hyper-parameter, the initial interval we consider is $[0.9, 0.999]$ as opposed to $[0, 0.99]$ (initial experiments always converged to a hyper-parameter in this interval - so we used this initial interval to reduce compute time).

Having found the optimal hyper-parameter $\lambda^\star$, we then train ten probers with $\lambda = \lambda^\star$ and random seeds. From these ten trained probers, we pick the prober that minimises the loss function (this is in line with the method used in Burns et al. [2022]). This is the prober that is used to compute the values found in the tables of this paper.

D Granular Data for Models and Specific Datasets

D.1 Accuracies

Dataset	Loss Function							
	CCS	MD-CCS	MD-Acc	MA	SMR	PCA	Rand.	Superv.
IMDB	0.9200	0.9225	0.9200	0.9225	0.9200	0.9220	0.7908	0.9300
Amazon	0.9438	0.9469	0.9469	0.9469	0.9469	0.9469	0.7344	0.9469
BoolQ	0.5225	0.5167	0.7825	0.7900	0.7925	0.7800	0.6047	0.8875
AG News	0.5275	0.5375	0.5300	0.5275	0.5275	0.5150	0.5152	0.9500
RTE	0.5175	0.5275	0.5275	0.5125	0.5225	0.5275	0.5365	0.7050
Average	0.6863	0.6902	0.7414	0.7399	0.7419	0.7383	0.6363	0.8839

Table 3: We compare test accuracies of different loss functions using the last hidden state of the encoder of UnifiedQA T5-Large for a number of datasets.

Dataset	Loss Function							
	CCS	MD-CCS	MD-Acc	MA	SMR	PCA	Rand.	Superv.
IMDB	0.9275	0.9250	0.9200	0.9325	0.9325	0.9150	0.7190	0.9200
Amazon	0.9400	0.9425	0.9450	0.6050	0.5900	0.7675	0.6355	0.9425
BoolQ	0.9775	0.9625	0.9625	0.9800	0.9875	0.8725	0.6303	0.9850
AG News	0.5924	0.5300	0.5300	0.5675	0.5325	0.5175	0.5750	0.9550
RTE	0.7150	0.7400	0.7325	0.6900	0.6875	0.6900	0.5833	0.7675
Average	0.8305	0.8200	0.8180	0.7550	0.7460	0.7525	0.6286	0.9140

Table 4: We compare test accuracies of different loss functions using the last hidden state of the decoder of UnifiedQA T5-Large for a number of datasets.

Dataset	Loss Function							
	CCS	MD-CCS	MD-Acc	MA	SMR	PCA	Rand.	Superv.
IMDB	0.6225	0.7050	0.7650	0.7400	0.5225	0.7435	0.5525	0.8225
Amazon	0.5950	0.6200	0.6200	0.6175	0.7100	0.5725	0.6422	0.8800
BoolQ	0.5075	0.5350	0.53375	0.5075	0.5075	0.5225	0.5360	0.5800
AG News	0.5250	0.5075	0.5150	0.5375	0.5325	0.5175	0.5478	0.9500
RTE	0.5050	0.5100	0.5150	0.5075	0.5050	0.5125	0.5232	0.5575
Average	0.5510	0.5755	0.5898	0.5820	0.5555	0.5737	0.5603	0.7580

Table 5: We compare test accuracies of different loss functions using the last hidden state of the decoder of GTP-Neo for a number of datasets.

Dataset	Loss Function							
	CCS	MD-CCS	MD-Acc	MA	SMR	PCA	Rand.	Superv.
IMDB	0.9550	0.9500	0.9475	0.9550	0.9475	0.9500	0.8067	0.9550
Amazon	0.9475	0.9425	0.9475	0.9400	0.9500	0.9425	0.8145	0.9500
BoolQ	0.6600	0.7475	0.8100	0.8125	0.8125	0.8100	0.7100	0.8275
AG News	0.5000	0.5000	0.8250	0.7700	0.7300	0.8250	0.6545	0.9425
RTE	0.8075	0.7875	0.8375	0.8475	0.8525	0.775	0.6585	0.8925
Average	0.7740	0.7855	0.8735	0.8650	0.8585	0.8605	0.7288	0.9135

Table 6: We compare test accuracies of different loss functions using the last hidden state of the encoder of DeBERTa for a number of datasets.

D.2 Cosine Similarities

Dataset	Loss Function							
	CCS	MD-CCS	MD-Acc	MA	SMR	PCA	Rand.	Superv.
IMDB	0.9305	0.7065	0.2929	0.2064	0.3167	0.2930	0.0214	0.3862
Amazon	0.9635	0.8177	0.3149	0.2777	0.3867	0.3150	0.0231	0.7639
BoolQ	0.8082	0.8000	0.0187	0.0332	0.0442	0.0194	0.0181	0.0860
AG News	0.5081	0.4817	0.4758	0.1997	0.2009	0.1835	0.0267	0.0328
RTE	0.9691	0.7111	0.3954	0.0069	0.0468	0.3919	0.0215	0.0224
Average	0.8359	0.7034	0.2995	0.1448	0.1991	0.2406	0.0222	0.2583

Table 7: We compute the average cosine similarities of the directions found using different loss functions to the directions of 20 CCS probers. We use the last hidden state of the encoder of UnifiedQA T5-Large for a number of datasets.

Dataset	Loss Function							
	CCS	MD-CCS	MD-Acc	MA	SMR	PCA	Rand.	Superv.
IMDB	0.9905	0.7111	0.4048	0.3867	0.3815	0.2378	0.0212	0.6790
Amazon	0.8357	0.6021	0.3969	0.0387	0.0076	0.0760	0.0165	0.7499
BoolQ	0.9850	0.8694	0.5393	0.2768	0.2808	0.1158	0.0194	0.8640
AG News	0.5924	0.5348	0.4547	0.0109	0.4274	0.3548	0.0254	0.0219
RTE	0.9899	0.9169	0.8558	0.1303	0.1187	0.1115	0.0314	0.6924
Average	0.8787	0.7269	0.5303	0.1687	0.2432	0.1792	0.0228	0.6014

Table 8: We compute the average cosine similarities of the directions found using different loss functions to the directions of 20 CCS probers. We use the last hidden state of the decoder of UnifiedQA T5-Large for a number of datasets.

Dataset	Loss Function							
	CCS	MD-CCS	MD-Acc	MA	SMR	PCA	Rand.	Superv.
IMDB	0.6296	0.2483	0.0785	0.0549	0.0367	0.0624	0.0212	0.3465
Amazon	0.3005	0.3059	0.3066	0.0092	0.0359	0.1351	0.0266	0.2310
BoolQ	0.6478	0.6683	0.6628	0.0175	0.0175	0.3581	0.0213	0.0136
AG News	0.3644	0.4627	0.3123	0.0133	0.0123	0.1781	0.0298	0.0531
RTE	0.6964	0.7296	0.7218	0.0183	0.1400	0.2166	0.0237	0.0295
Average	0.5277	0.4830	0.4164	0.0226	0.0485	0.1901	0.0245	0.1347

Table 9: We compute the average cosine similarities of the directions found using different loss functions to the directions of 20 CCS probers. We use the last hidden state of the decoder of GPT-Neo for a number of datasets.

Dataset	Loss Function							
	CCS	MD-CCS	MD-Acc	MA	SMR	PCA	Rand.	Superv.
IMDB	0.9376	0.5504	0.5466	0.2599	0.1685	0.0554	0.0143	0.6811
Amazon	0.9538	0.2760	0.2406	0.1657	0.2401	0.0648	0.0178	0.8104
BoolQ	0.7823	0.7377	0.0357	0.0448	0.0543	0.0358	0.0175	0.2912
AG News	0.7003	0.6401	0.0145	0.1750	0.0140	0.0146	0.0222	0.0435
RTE	0.9476	0.9002	0.5557	0.5093	0.5349	0.1998	0.0294	0.4823
Average	0.8643	0.6209	0.2786	0.2309	0.2024	0.0741	0.0202	0.4617

Table 10: We compute the average cosine similarities of the directions found using different loss functions to the directions of 20 CCS probers. We use the last hidden state of the encoder of DeBERTa for a number of datasets.

D.3 Results from Hyper-Parameter Grid Search

We find that the MD-CCS hyper-parameter is always close to 1. This suggests that the trade-off most similar to CCS is one that prioritises minimising σ_m^2 (which has a coefficient of at least 0.9) over maximising σ_d^2 (which has a coefficient of at most 0.1).

For the MD-Acc loss function, we find that the hyper-parameter is often either close to 0 or close to 1. When the hyper-parameter is close to 1, the MD-CCS and MD-Acc have landed on the same hyper-parameter. On the other hand, when MD-Acc has a λ is close to 0, the loss function is very similar to PCA.

Dataset	Loss Function			
	MD-CCS	MD-Acc	MA	SMR
IMDB	0.9891	0.0000	0.7524	0.6930
Amazon	0.9812	0.0000	0.6930	0.5940
BoolQ	0.9752	0.0000	0.0000	0.0000
AG News	0.9871	0.9801	0.0000	0.0000
RTE	0.9891	0.0990	0.9900	0.1980

Table 11: Hyper-parameters values for the encoder of UnifiedQA T5-Large.

Dataset	Loss Function			
	MD-CCS	MD-Acc	MA	SMR
IMDB	0.9871	0.8910	0.6732	0.4950
Amazon	0.9653	0.7128	0.6930	0.6930
BoolQ	0.9535	0.7920	0.5940	0.5544
AG News	0.9812	0.8910	0.9009	0.0000
RTE	0.9000	0.8118	0.3762	0.1980

Table 12: Hyper-parameters values for the decoder of UnifiedQA T5-Large.

Dataset	Loss Function			
	MD-CCS	MD-Acc	MA	SMR
IMDB	0.9891	0.9900	0.5940	0.4950
Amazon	0.9891	0.8910	0.4950	0.6534
BoolQ	0.9752	0.0000	0.0000	0.0990
AG News	0.9891	0.0000	0.0000	0.8910
RTE	0.9792	0.3366	0.3168	0.2772

Table 13: Hyper-parameters values for the encoder of DeBERTa.

Dataset	Loss Function			
	MD-CCS	MD-Acc	MA	SMR
IMDB	0.9713	0.0990	0.5742	0.4950
Amazon	0.9911	0.9900	0.5940	0.5940
BoolQ	0.9871	0.9900	0.9108	0.4950
AG News	0.9871	0.8910	0.9702	0.9900
RTE	0.9871	0.9900	0.9900	0.5940

Table 14: Hyper-parameters values for the decoder of GPT-Neo.