DICTIONARY LEARNING FOR DISENTANGLING CONCEPTS IN VISION TRANSFORMERS.

Anonymous authors
Paper under double-blind review

ABSTRACT

Dictionary learning for the activations of vision transformer models enable new insights based in Mechanistic Interpretability by extracting concept-based encodings for model activations. In this work propose mapping the activations of each layer to their sparse concept-based encoding using learned dictionaries, and construct a "sparse graph representation" using simple logistic regression with an l_1 weight penalty on binarized concept labels. We analyze wether such a graph can faithfully model the inner workings of a vision transformers, and discuss potential applications for analyzing inherent biases, causal reasoning directions, and adversarial defense guarantees.

1 INTRODUCTION

The transformer architecture (Vaswani et al. 2017) is being used for a wide range of applications, including computer vision (Dosovitskiy et al. 2021) and natural language processing. Although it is commonly assumed that transformer models can do some amount of reasoning, due to their high-dimensional nature, it is difficult to verify exactly how the reasoning steps go.

However, recently progress has been made to *disentangle* inner activations into a sparse set of discrete "concepts" (Bricken et al. 2023). In this setting, a specific activation or embedding is represented by a combination of few "dictionary vectors" which are aggregated (e.g. through a weighted sum). Currently, it remains difficult to describe these "concepts" in human-compatible terms. Still, the low-dimensional nature of the concept-based representation may allow further insights about the "reasoning steps" that a model makes.

In this work, we try to illuminate this question further for the case of a vision transformer. In particular, we first review how we can "reveal" the concept-based representation using either sparse autoencoder (SAE) or classical dictionary learning (K-SVD) techniques. Then, we discuss how we can use the concept-based representation to predict the $(l+1)$th layer from the (l)th layer. Finally, we try to model an entire 12-layer vision transformer by using only our extracted "prediction graph", and compare the quality of the prediction graph.

We suggest that successful representation of a vision transformer through a concept-based graph unlocks a series of advantages:

Interpretability and Alignment: If concepts even on the internal layers are well understood, the concept-based graph allows for strong insights about the model's reasoning, and the alignment of the reasoning chain to our societal values. For example, it may be wrong and/or unwanted for the model to make conclusions about a human's credit score rating based on their ethnicity. By constructing the proposed graph, however, such dependencies could be caught, and mitigated.

Verification of Causal Direction: In a similar vein, the model may make assumptions about the causal direction of concepts or statistical variables. While the causal direction is typically not important for statistical inference, it does become important when trying to generalize to unseen inputs (see e.g. Schölkopf (2022)).

Sensible Regulation: While asserting the correct representation of societal values and causal directions may be of interest to practitioners, it also provides a direct avenue for regulators

to propose sensible regulation, that e.g. takes the form "there must be no causal reasoning edge between ethnicity and credit score". Such assertions were previously difficult to verify.

Adversarial Testing: The low-dimensional graph structure may also enable new formal verification approaches to such transformer networks, by constructing a adversarial robustness guarantee directly on the concept-graph, rather than on the full machine learning model.

1.1 Related work.

- world representations: (Li et al. 2022)
- Bayesian Networks (e.g. (Kochenderfer, Wheeler, and Wray 2022))
  - however, difficult to get to work due to the size of the problem, and typically boils down to some kind of greedy search and tabular lookup. Still might be worth a shot though.

1.2 Fundamental hypothesis

- data is truly coming from these generative factors, and only a small number is active at any moment.
- we can find one basis vector per generative factor.

2 Mathematical Background.

2.1 Dictionary Learning for Disentanglement.

Higgins et al. (2017) introduced the concept of "disentanglement" where data is generated through a ground-truth simulation process based on a number of generative factors, which are (i) conditionally independent, and (ii) interpretable. The goal of disentanglement is therefore to recover these generative factors from the data. Locatello et al. (2019) showed that in the general setting and without further assumptions, it is impossible to find the true generative factors. However, several promising advances have been made, e.g. by exploiting the "arrow of time" ((Dunion et al. 2022)) or additionally requiring sparsity (e.g. (Bricken et al. 2023)).

We will focus on the latter, i.e. disentanglement through dictionary learning and sparse coding. We consider a set of (high-dimensional) embedding vectors $Z = \{z_i\}_{1 \leq i \leq |Z|}$ with $z_i \in \mathbb{R}^n$. Dictionary Learning then aims to find a set of new basis vectors $D = \{d_i\}_{1 \leq i \leq |D|}$ with $d_i \in \mathbb{R}^n$ and $|D| > n$, together with a sparse encoding matrix $X \in \mathbb{R}^{n \times |Z|}$, such that $Z \approx DX$. In other words, any embedding vector z can be approximated by summing up a small number of (weighted) basis vectors from D, i.e.

$$z \approx \sum_{(\lambda, i) \in \text{nonzeros}(x)} \lambda \cdot d_i$$

for any z, where z and x are a corresponding columns of Z and X.

Such dictionaries and encoding matrices can for example be constructed through a sparse autoencoder with an affine encoder and affine decoder, trained on a reconstruction loss together with a l_1-penalty on the parameters (Bricken et al. 2023), or through an iterative algorithm like K-SVD (Aharon, Elad, and Bruckstein 2006).

2.1.1 write comment on using (O)MP to find x from D and z.

2.2 Logistic Regression for Imbalanced Classes, and the ROC curve and AUC characteristic.

We will need a simple model to make predictions whether a certain concept is active in layer $(l+1)$ based on the information from layer (l). We choose a simple logistic regression model, i.e. $f(x) = \sigma\left(\theta^\top x\right) \in (0, 1)$ (with a bias) where $\sigma(\xi) = \frac{1}{1+e^{-x}}$ denotes the sigmoid function.

Such models may be trained by minimizing e.g. $\sum_{(x,y)\in\mathcal{D}} bce\left(f(x), y\right) + \mu\|\theta\|_1$ for dataset $\mathcal{D}$, where $bce(\cdot)$ denotes the binary cross-entropy and the parameter μ acts as a regularizer that encourages sparsity. However, in the scenario of a strongly unbalanced dataset, i.e. where the number of $y = \top$ (true) vastly outnumbers $y = \bot$ (false) or vice versa, a correction must be taken both for training and evaluation. One possibility is a resampling based approach, where a new dataset $\mathcal{D}'$ is constructed by sampling (with replacement) data from the original dataset in a weighted manner, such that the number of $y = \top$ and $y = \bot$ are approximately the same.

Finally, in order to evaluate the predictor we use the Receiver-Operator-Characteristic (ROC) curve, which plots the false positive against false negative rate at different threshold levels τ with $y = [f(x) > \tau]$, and report the Area-Under-Curve (AUC) metric.

2.3 Vision Transformers.

Transformer models typically operate on a tuple of *tokens* $T = (t_k)_{1\leq k\leq |T|}$ with $t_k \in \mathbb{R}^n$ and $T \in \mathcal{T}$ that encode some information. In particular, the models are typically made up of a series of transformer layers $model(\text{image}) = (l_{n_{\text{layers}}} \circ l_{n_{\text{layers}}-1} \circ \cdots \circ l_1)(tokenizer(image))$ where $\circ$ denotes function composition, and each layer has transforms a tuple of tokens to another tuple of tokens, i.e. $l_i : \mathcal{T} \to \mathcal{T}$. Finally, for vision models, the first token in the tuple is the "classification token" and has been shown to be useful for solving a variety of downstream tasks (citation needed). Let us denote by z_1 the class token of $l_1(tokenizer(image))$, by z_2 the class token of $(l_2 \circ l_1)(tokenizer(image))$ and so forth.

3 Proposed Method.

We will now consider how to construct the concept-based graphical representation to approximate the inner workings of a vision transformer. The rough order of operations will be as follows:

(i) Collect class tokens (given image dataset and vision transformer model);

(ii) Run dictionary learning algorithm on each layer's class tokens;

(iii) Consider the class token encodings from adjacent layers;

  (a) For each concept of the "output" layer, select features to use from the "input" layer;

  (b) For each concept of the "output" layer, train a classifier using the selected features from the "input" layer;

(iv) Repeat for each pair of adjacent layers;

(v) Run inference through repeated sampling.

Section 2.3

In the following we describe each step in more detail. Experimental results follow in Section 4.

3.1 Collecting class tokens.

As introduced in Section 2.3, we evaluate the model on a series of input images, and store the class token activation of each layer, i.e. $z_1, z_2, \ldots, z_{n_{\text{layers}}}$ for each image. Let us write $Z_1 = \left[z_1^{\text{img}=1} z_1^{\text{img}=2} \ldots z_1^{\text{img}=|Z|}\right]$ to be a matrix of concatenated class token activations (after the first layer) for all images; and similarly for Z_2, Z_3 etc.

Notice again that it has been shown that the last set of class tokens $Z_{n_{\text{layers}}}$contains sufficient information for many downstream tasks, even if it has not been explicitly trained for that (see e.g. He et al. (2021)).

3.2 BUILDING THE DICTIONARIES.

Next, we run the dictionary learning algorithm once for each layer. In particular, following Section 2.1, we construct the tuple $\left(D_l, \tilde{X}_l\right) = dictionarylearning(\tilde{Z}_l)$ for each layer $1 \leq l \leq n_{\text{layers}}$, with $\tilde{Z}_l \subseteq Z_l$, i.e. we may use a subset of all embeddings.

For dictionary learning, we have two parameters to choose: (i) the number of dictionary elements $|D|$, and (ii) either the number of nonzero elements per sample (K-SVD) or the strength of the l_1 penalty (Sparse Autoencoder).

For (i), notice again that the number of dictionary elements $|D|$ is typically chosen to be greater than the dimensionality of the embeddings n, which makes sense only if additional sparsity is required (otherwise, data in $\mathbb{R}^n$ needs at most n basis vectors). Indeed, following Section 1.2, we would need one basis vector per "generative factor" or "underlying concept". It is therefore clear that the choice of $|D|$ is not just a numerical choice, but heavily depends on the underlying data.

We believe that there is more work to be done here. However, for our experiments we choose $|D| = 4n = 3072$ which follows the suggestions of (Bricken et al. 2023).

A similar situation occurs for (ii): The number of nonzero elements in any sample's encoding x directly corresponds to the number of "active" concepts for a given input. Therefore, a simple image may have only one or few nonzeros, whereas a "busy" or otherwise complex image may have many.

For our experiments we choose to restrict the number of nonzero values in each encoding x to be $nnz(x) \leq 10$.

3.3 EXTRACTING THE GRAPH FOR ADJACENT LAYERS.

Let's consider now two adjacent layers l_{left} and l_{right} of the vision transformer, and their corresponding dictionaries D_{left} and D_{right} constructed above.

Notice that we can use these dictionaries to encode any new activations using sparse coding (see Section 2.1), i.e. $z_{\text{dir}} \approx D_{\text{dir}} x_{\text{dir}}$ where x_{dir} has few nonzero elements.[1] Let us denote any specific feature activation i by $x^{[i]}_{\text{dir}}$, and further let $y^{[i]}_{\text{dir}} = [x^{[i]}_{\text{dir}} \neq 0] \in \{\top, \bot\}$ indicate whether a specific feature i is active (with any nonzero value) or not.[2]

Our goal is then to predict $y^{[i]}_{\text{right}}$ using a small subset of features from x_{left}.

To proceed, we must therefore discuss three questions:

1. choice of prediction model;
2. approach for feature selection
3. dealing with imbalanced data

For the prediction model, we choose a simple logistic regression model with l_1 regularization on the parameters to model $y^{[i]}_{\text{right}} = f_i(x_{\text{left}})$, as described in Section 2.2. The l_1 regularization forces the model to select only a limited number of inputs, leading to feature selection. However, notice that our labels $y^{[i]}_{\text{right}}$ are extremely unbalanced, in the sense that they are true ($\top$) only approximately $\frac{n_{\text{nnz}}}{k} = \frac{10}{3072} \approx 0.3\%$ of the time in our case. To deal with this, we resample the dataset in order to rebalance the classes, and further create an ensemble with $T = 20$ of such undersampling learners. This approach follows e.g. Liu, Wu, and Zhou (2009).

Finally, we find those features $x^{[j]}_{\text{left}}$ which are "used" in at least 80% of the constructed models, and use those to create the actual classifier on another rebalanced dataset.

The algorithm is summarized below.

```
function find_single_node_predictor(
```

[1] Read "dir" as "direction", i.e. either "left" or "right".

[2] Here, $[\cdot]$ denote the Iverson bracket.

```
    X::Vector{Vector},
    y::Vector{Bool};
    T = 20,
    l1_penalty=1.0
)
  machine = LogisticClassifier(l1_penalty)
  models = map(1:T) do t
      X_sub, y_sub = resample(X, y; replace=true, rebalance=true)
      return fit(machine, X_sub, y_sub)
  end

  compute_feature_usage_rate(feature_idx) =
      mean(models) do model
        weights(models[t])[feature_idx] != 0
      end

  selected_features = findall(i->feature_usage_rate(i) > 0.8,
                              1:n_features)

  X_selected = [x[selected_features] for x in X]
  final_model = fit(machine, X_selected, y)
end
```

Figure 1: Algorithm for constructing a single concept node predictor $Pr\left(y_{\text{right}}^{[i]}\right) = f_i(x_{\text{left}})$.

3.3.1 USING FULLY BINARIZED SAMPLES.

The approach introduced above maps x_{left} to $y_{\text{right}}^{[i]}$, i.e. a vector in $\mathbb{R}^n$ to a binary label. However, for full model inference it will be convenient to work in the binary space only, i.e. mapping y_{left} to $y_{\text{right}}^{[i]}$.

3.4 RUNNING INFERENCE.

To run full model inference, we first construct a binary predictor $y_{\text{left}} \rightarrow y_{\text{right}}^{[i]}$ for each pair of adjacent layers, and each i in $1 \leq i \leq n$. Then, we can first run the *tokenizer* on the input, and then run all the models for the first layer to compute the probabilities of each active concept of the second layer.

Here, we can either use a sampling-based approach, or a maximum-prediction-based approach to decide on the activations of the next layer. Now we apply the next series of models, and so forth, until we have predicted the concept activations of all layer's activations.

4 EXPERIMENTAL RESULTS.

4.1 SETUP.

We consider the *sbucaptions* dataset (Ordonez, Kulkarni, and Berg 2011) which contains about one million captioned images, of which we randomly sample about one hundred thousand. We process each image with the vision transformer `clip-vit-base-patch32` (Radford et al. 2021), which has 12 layers, and find the dictionary with the K-SVD algorithm[3]

When extracting the graph, we choose the l_1 penalty as $\mu = 5.0$.

For now, we only present results on the last layer. We will first analyze the number of "active" features selected by the algorithm, and then the prediction performance.

[3] `https:github.com/RomeoV/ksvd.jl`

4.2 NUMBER OF PREDICTION FEATURES.

Let us consider the predictors $f_i(x_{11})$ (or $f_i(y_{11})$) for the concept activations of the last layer $y_{12}^{[i]}$. Given $\mu = 5.0$, we analyze the number of chosen features to make each prediction, and plot the results in Fig. 2.

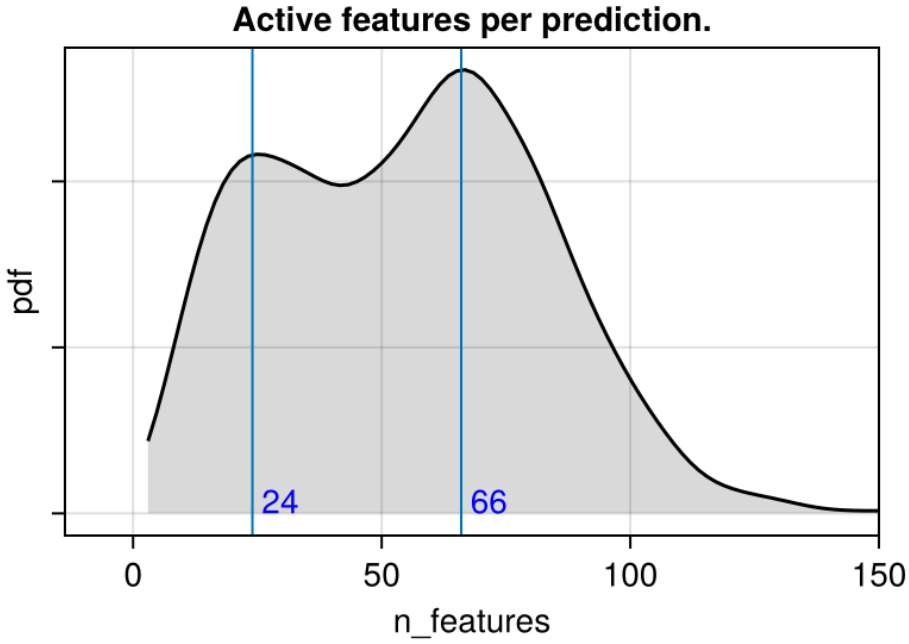

Figure 2: The number of "parent" concepts to predict each concept activation of the last layer (i.e. embedding).

First, we observe that indeed almost all features can be predicted using less than 100 parent concepts each, out of a possible $4 \cdot 768 = 3072$ parents. Therefore, we conclude that we have found a much more "sparse" representation of the underlying reasoning graph.

Further, we can see that there are two peaks in the graph, one at 24 parents and one at 66. We suggest two possible explanations:

(I) There are "simplistic" and "complex" features. Simplistic features can be predicted using a smaller number of parents, and vice versa;

(II) there are "well disentangled" and "poorly disentangled" concepts (either inherent to the data/domain or due to our algorithm). Poorly disentangled concepts are therefore not represented correctly, and have to rely on many more predictors to achieve good prediction accuracy.

4.3 PREDICTION PERFORMANCE FOR A SINGLE LAYER.

In order to investigate the prediction performance, recall again that we have very unbalanced class labels, with only about 0.3% of the labels being true. Therefore, we present the Receiver-Operator-Characteristic Curve (ROC Curve) in Fig. 3, as well as the Area-Under-Curve (AUC) metric.

We can see the vast majority of curves have a prediction accuracy significantly outperforming random guessing. The distribution of AUCs for all $y_{\text{right}}^{[i]}$ is 0.897 ± 0.073 (one std) and median 0.914. For visualization purposes, we show a "virtual" ROC curve in red with the median AUC.[4]

[4]Here, we're using the model function $f_k(x) = 1 - \left((1-x^k)/(1+x^k)\right)^{\frac{1}{k}}$ with $k = 0.492$.

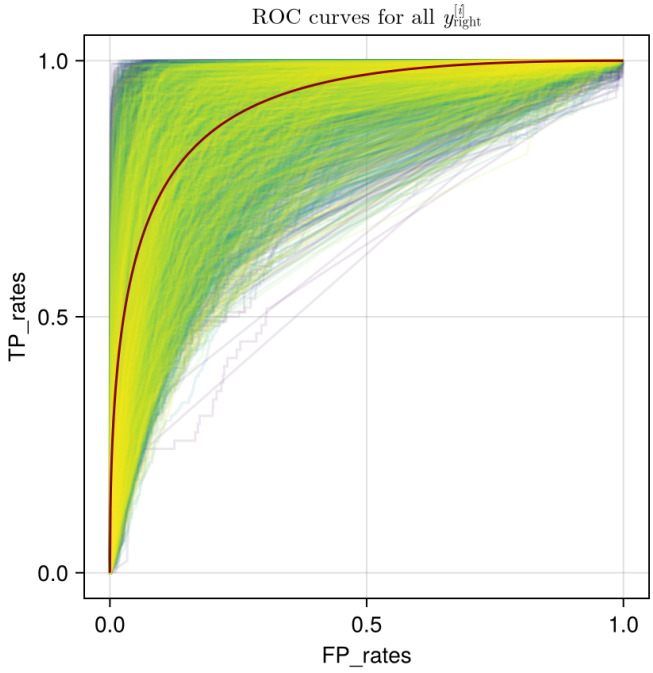

Figure 3: Receiver-Operator-Characteristic curve for all concept predictors from layer $11 \rightarrow 12$.

We also compute the F_1-score for each $y_{\text{right}}^{[i]}$ when simply choosing the prediction with higher likelihood, i.e. threshold $\tau = 0.5$. Across all i, we get an F_1-score with $0.84 \pm 0.09\sigma$ and median 0.86.

4.4 Next steps:

Next, we want to construct the layer-wise predictors for all layers, and evaluate the above metrics. Then, we want to evaluate the sampling strategies, and see if we can make a full network inference using the binarized concept labels.

Finally, we also suggest that we can do a "abstract interpretation" analysis of the full network, as well as counterfactual experiments, by modifying the concept labels directly.

5 BIBLIOGRAPHY:

Aharon, M., M. Elad, and A. Bruckstein. 2006. "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation." *Ieee Transactions on Signal Processing*, no. 11 (November).

Bricken, Trenton, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, et al. 2023. "Towards Monosemanticity: Decomposing Language Models with Dictionary Learning." *Transformer Circuits Thread*.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, et al. 2021. "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale." arXiv. https://arxiv.org/abs/2010.11929.

Dunion, Mhairi, Trevor McInroe, Kevin Sebastian Luck, Josiah P. Hanna, and Stefano V. Albrecht. 2022. "Temporal Disentanglement of Representations for Improved Generalisation in Reinforcement Learning."

He, Kaiming, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2021. "Masked Autoencoders Are Scalable Vision Learners." December 19, 2021. https://arxiv.org/abs/2111.06377.

Higgins, I., L. Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, M. Botvinick, S. Mohamed, and Alexander Lerchner. 2017. "Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework." In *ICLR*.

Kochenderfer, Mykel J., Tim A. Wheeler, and Kyle H. Wray. 2022. *Algorithms for Decision Making*. MIT Press.

Li, Kenneth, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2022. "Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task." October 25, 2022. https://arxiv.org/abs/2210.13382.

Liu, Xu-Ying, Jianxin Wu, and Zhi-Hua Zhou. 2009. "Exploratory Undersampling for Class-Imbalance Learning." *Ieee Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, no. 2 (April).

Locatello, Francesco, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. 2019. "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations." In *Proceedings of the 36th International Conference on Machine Learning*. PMLR.

Ordonez, Vicente, Girish Kulkarni, and Tamara Berg. 2011. "Im2Text: Describing Images Using 1 Million Captioned Photographs." In *Advances in Neural Information Processing Systems*. Curran Associates, Inc.

Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, et al. 2021. "Learning Transferable Visual Models From Natural Language Supervision." In *Proceedings of the 38th International Conference on Machine Learning*. PMLR.

Schölkopf, Bernhard. 2022. "Causality for Machine Learning." In *Probabilistic and Causal Inference: The Works of Judea Pearl*, 1st ed. Association for Computing Machinery.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need." In *Advances in Neural Information Processing Systems*. Curran Associates, Inc.