DISENTANGLING CONCEPTS AND CONSTRUCTING HUMAN-COMPATIBLE EXPLANATIONS FOR IMAGE EMBEDDINGS

Romeo Valentin and others

ABSTRACT

Vision Transformers produce high-quality embeddings useful for a variety of downstream tasks – although both their embeddings and inner workings are typically difficult to interpret. By disentangling the conceptual components of the data we can hope to gain additional insight. Finding such disentangled human-aligned conceptual components has been shown to be difficult in the past. In this work we propose using the K-SVD algorithm together with captioned image datasets to disentangle the final embeddings and assign human-compatible "explanations" for each of the disentangled "concepts". To achieve this, we present a new highly optimized implementation of the K-SVD algorithm that scales to large datasets with millions of samples and can be scaled to many compute nodes. We find that we can successfully cluster images by their active concepts and that the explanations hold up to visual inspection, but that it is difficult to quantitatively measure the quality of the explanations.

1 INTRODUCTION.

The Transformer architecture has been successfully deployed for a variety of applications. However, as with other machine learning methods, its reliability is not guaranteed. Reliability can for example be measured as a combination of in-distribution performance, performance under distributional shift, out-of-distribution detection and/or performance, uncertainty calibration, and robustness to adversarially chosen input perturbations.

We hypothesize that for robustness and generalizability, the model must learn a faithful representation of the world, which goes beyond mere statistical inference.

Humans similarly have a model of the world, and although it is generally not easy to write down, we have developed natural language as a way to communicate information between each other. It is therefore natural to wonder whether the models potentially emerging in large Transformer networks are or can be aligned with a human model of the world.

We consider the special case of Vision Transformers. Typically, models are trained to produce a sequence of high-dimensional embeddings, which can serve as inputs to downstream tasks like classification. It is therefore clear that these embeddings have a variety of information stored in them.

The goal is now try to "disentangle" these embeddings in order to provide a interpretable/human-aligned/monosemantic view of the embeddings, which stay faithful to the original content of the embedding (but might sacrifice some of the detailed information). This goal of "disentanglement" from a "block of data" goes back to Independent Component Analysis (ICA), and was picked up again in the context of machine learning (see e.g. beta-VAE (Higgins et al. 2017)). "Disentanglement" here is defined as having a set of components that are both conditionally independent and align with a notion of human interpretability. However, an important result in 2020 proved that without further assumptions there is an infinite set of components which satisfy the conditional independence property, so requiring this alone is not enough to get interpretable components for free (Locatello et al. 2019).

However, in this work we propose two extra conditions with which we aim to overcome the theoretical concerns:

1. small number of "active" concepts, and
2. encodable using natural language.

1.1 Small number of "active" concepts.

The first condition is motivated easily. It turns out to be easy to project a dataset onto a new basis of orthogonal vectors, i.e. "statistically independent concepts", so that each datapoint is a sum of these orthogonal vectors – this is indeed just the singular value decomposition (SVD), i.e. $Y = U\Sigma V^\top$ where each column of Y is a datapoint, and each column of U is a "concept vector".

However, in this case each datapoint is constructed of an "infinite sum" of concept vectors, whereas we believe that any data sample only has a relatively small number of active concepts. Further, the SVD is designed to find something like "Principal Components", which in our language would translate to abstract concepts that are represented in the majority of datapoints. However, we want neither of these. Instead, we would like to represent each datapoint y for example as a sum $\sum_{j\in\mathcal{J}} \vec{d}_j \cdot \lambda_j$ where $|\mathcal{J}|$ is small. In other words, we try to write our data matrix $Y = [y_1 \dots y_{|Y|}]$ as a product of dictionary vectors $D = [d_1 \dots d_{|D|}]$ and a sparse assignment matrix X such that $Y \approx DX$.[1]

It turns out that this problem has been studied previously, and is known as "Dictionary Learning". A popular algorithm for dictionary learning is the "K-SVD" algorithm (Aharon, Elad, and Bruckstein 2006), which essentially alternates between updating the set of dictionary vectors and updating the assignment matrix. We will go into more details in Sec. sec:ksvd-algorithm.

Recently, it has also been proposed to use a Sparse Autoencoder, e.g. as presented in (Bricken et al. 2023). We briefly compare the approaches in Sec. subsec:ksvd-vs-sparse-autoencoder.

1.2 Natural language encoding.

- The additional sparsity constraint is enough to overcome the theoretical problems brought up by Locatello et al. (2019).
- However, it remains unclear whether this is enough for extracting dictionary or concept vectors with the kind of human-aligned interpretability that we are looking for.
- Indeed, even if they were sufficiently aligned, it is not clear how we can map the concept vectors to natural language representations, or how we can test whether a language-to-embedding mapping is "faithful" or correct in some sense.
- However, we believe that the recently gained availabily to large language models (LLMs) gives us a way to overcome these problems.

Proposition 1: *Consider the setup presented in Fig. fig:commutative-diagram, and consider that the disentangling method works well in the sense described in the previous section. Then, we consider a token mapping "faithful" if the diagram in Fig. fig:commutative-diagram commutes.*

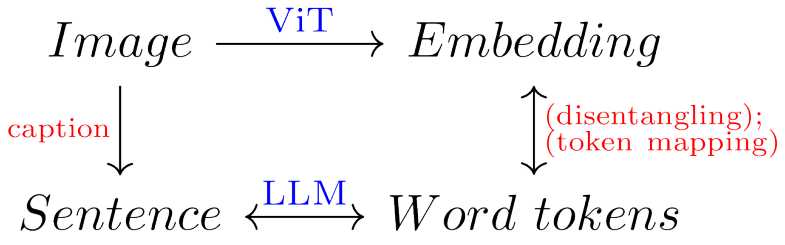


Figure 1: Commutative diagram showing the flow of semantic information. Given captioned images, a ViT, an LLM, and a disentangling method, we can therefore create a token mapping.

[1] Note that having a linear dependence of Y wrt D is a design choice, and one may also consider more nonlinear relations. However, for tractability we restrict ourselves to the linear setting.

Being able to "score" a given token mapping now may make it possible to actually construct the token mapping.

2 PRELIMINARIES.

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).

2.2 IMAGE CAPTIONING EVALUATION.

In image captioning evaluation we consider a generated image caption and assign a score either reference-based, i.e. with respect to a set of other captions, or reference-free, i.e. only using the image. Hessel et al. (2021) have proposed the `CLIPScore` that computes the cosine distance between embeddings of the image and caption, where the embeddings are generated by a multimodal model such as CLIP (Radford et al. 2021), which notably must also have been trained by minimizing embeddings cosine distances or similar metrics (Steck, Ekanadham, and Kallus 2024).

Hessel et al. (2021) find that in this metric scores typically do not exceed 0.4, even for highly similar texts, nor fall under 0. We will see a similar patterns in our results.

2.3 DICTIONARY LEARNING WITH THE K-SVD ALGORITHM.

In this section we first discuss some theoretical properties and considerations of the K-SVD algorithm, and then discuss the contributions we have made and published in the KSVD.jl package. Then we will briefly discuss the K-SVD algorithm versus using a Sparse Autoencoder.

2.3.1 THE ALGORITHM, AND THEORETICAL CONSIDERATIONS.

The K-SVD algorithm solves the problem of dictionary learning outlined above, namely finding a matrix decomposition $Y \approx DX$ where $\text{size}(D, 2) > \text{size}(D, 1)$ (i.e. an "overcomplete" dictionary), and X is sparse. It can be understood as a generalization to the k-means algorithm, but allowing any datapoint to be associated with *multiple centroids*.

Typically, iterative dictionary learning algorithms alternate between two steps:

Sparse coding: Given a fixed dictionary, for each data sample find a small subset of dictionary vectors and factors such that $y_i \approx \sum_{j \in \mathcal{J}} x_j \cdot d_j$. Note that this is a non-convex problem

(and indeed NP-hard), and typically heuristics like greedy search are used. Typically used algorithms are (Orthogonal) Matching Pursuit, Basis Pursuit, or FOCUSS. In our implementation, we use Matching Pursuit.

Dictionary update: Given a coding matrix X, we can again update the dictionary elements. Several approaches exist – for example, a simple gradient descent based approach can be used to optimize $\min_D \|Y - DX\|$, where X stays fixed. However, Aharon, Elad, and Bruckstein (2006) propose updating each dictionary element independently by considering all the data points that are "using" a given dictionary element and then replacing the dictionary element with the dominant singular vector of the these datapoints. Note that this simultaneously updates the values in X, which most other algorithms do not.

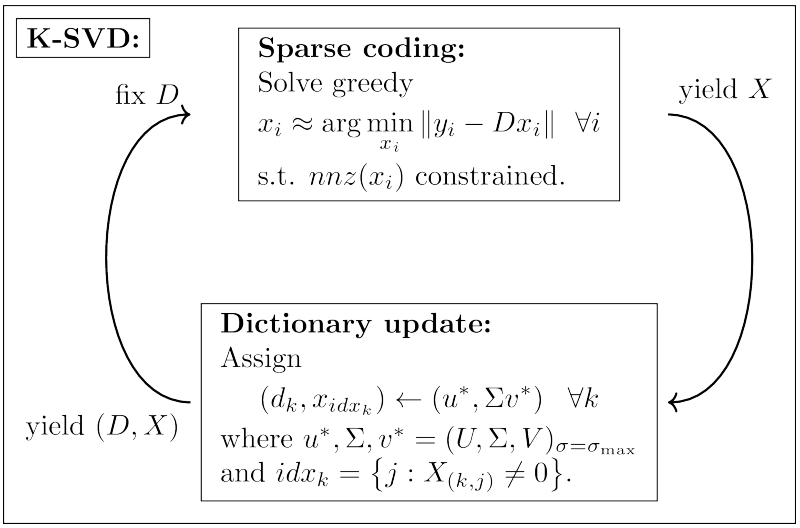


Figure 2: An overview of the K-SVD algorithm.

This algorithm has exceptionally few hyperparameters – indeed, it is sufficient to choose a number of dictionary elements, and a cutoff threshold for sparse coding.[2] A mere two hyperparameters! Further, utilizing the (truncated) svd algorithm typically leads to fast convergence and good computational efficiency, since the svd is theoretically "optimal" in some sense, and heavily optimized.

2.3.2 Dictionary Learning vs Sparse Autoencoder.

Method	k-svd	gradient
find sparse assignments via...	greedy search	L_1 loss
number of hyperparameters	1/2	many
"one" way to do it	yes	no
can get stuck in local minima?	yes (?)	yes (?)
runtime?	Comparison tbd...	
quality of result?	Comparison tbd...	

3 Proposed Method.

Our goal is to extract a set of *concepts* that make up the underlying structure of the images and embeddings. We will first describe how to extract these concepts as high-dimensional "basis vectors" of the embeddings, together with sparse assignment matrices, and then describe how we can assign human-compatible explanations to each of the concept, and measure the quality of the explanations. Finally, we briefly introduce `KSVD.jl`, our custom implementation of the K-SVD algorithm that scales up to millions of samples to solve the dictionary learning / concept finding problem.

[2] Alternatively, we can specify a maximum number of dictionary elements per data sample.

3.1 CONCEPT GENERATION THROUGH DICTIONARY LEARNING.

In the first step, we consider the embeddings generated by a vision transformer model and represent the embeddings as a sum of "dictionary elements" each of aims to represent a human-compatible "concept". In particular, we compute the embeddings of a set of input images, and then run a dictionary learning algorithm on the collected embeddings (see subsec:dictionary-learning), which yields a set of dictionary elements (vectors) $D = [d_1 d_2 \dots d_{n_{\text{dicts}}}]$ and the sparse assignment matrix $X = [x_1 x_2 \dots x_{n_{\text{samples}}}]$ with $Z \approx DX$, where each column of Z is the embedding vector of a single image. Then, we assume that each dictionary element d_i corresponds to one human-compatible concept.

3.2 EXPLANATION GENERATION THROUGH CAPTION AGGREGATION.

Having computed the set of dictionary elements in the previous step, we generate the human-compatible explanations for each concept as follows:

1. Consider a dataset of *captioned* images and compute the embeddings $\bar{z}$ for each image, and solve the sparse representation problem (subsec:dictionary-learning), which yields $\tilde{\bar{z}}$ and the λ_i.
2. To generate an explanation for the i-th dictionary element, we now collect those captions for which $\lambda_i \neq 0$, where $lambda_i$ was computed from the corresponding images. In other words, we collect the captions of all images for which atom i is active.
3. Finally, we use a large-language model to find a *common theme* in the captions. The theme may contain multiple words, but shall be concise.

3.3 EVALUATING THE EXPLANATIONS.

To measure the quality the explanations, we use them to generate image captions, and measure their quality with the `CLIPScore` introduced in subsec:CLIPScore. We generate the caption as follows: For a given image, we consider again the active concepts where $\lambda_i \neq 0$ and aggregate their "explanations". Then we concatenate the explanations as a simple list bullet point list of the explanations (in markdown format) and prefix the text with "A photo containing".

3.4 KSVD.JL : A HIGHLY OPTIMIZED K-SVD IMPLEMENTATION.

Bricken et al. (2023) mention the K-SVD algorithm for dictionary learning, however deem it computationally infeasible to apply to large datasets with millions (or billions) of samples. And indeed, current implementations seem not up to the task; the implementation available as `sklearn.decompositions.MiniBatchDictionaryLearning`, which extensively leverages `numpy` and `joblib`, takes over 3 minutes for ten iterations on a dataset with ten thousand elements (despite multi-threading).

For this reason, we present `KSVD.jl`, an implementation of the K-SVD algorithm in the Julia programming language (Bezanson et al. 2017). This implementation outperforms sklearn's implementation by about $50\times$ when computing the same problem, can gain an additional $2\times$ by reducing the precision from `Float64` to `Float32`, and can be scaled across many compute nodes with almost linear speedup improvements[3]. That means if, for example, eight compute nodes are available, we can expect a speedup of $(50 \cdot 2 \cdot 8)\times = 800\times$ for moderate to large datasets. Further, `KSVD.jl` also employs several algorithmic modifications that, to the author's knowledge, lead to faster convergence given the same number of compute operations.[4]

This speedup has been achieved through extensive benchmarking and optimization of the code, including

- careful adjustments to the execution order and small algorithmic adjustments,

[3] Unlike the previous two numbers, this hasn't been properly tested yet.

[4] A more detailed study on this is to follow.

- single-core optimizations like aggressive buffer preallocations, exploiting cache locality, improving the memory layout, and reducing memory movements,
- careful multi-threading using small batch updates with frequent cross-communication implemented with Julia's efficient task scheduling,
- a custom multi-threaded dense-sparse matrix multiplication implementation (`ThreadedDenseSparseMul.jl`),
- pipelined GPU-offloading for large matrix multiplications (currently unused in the fastest version),
- a custom distributed executor allowing to spread the computation over many compute nodes.

OVER-THE-THUMB ESTIMATION OF COMPUTATIONAL REQUIREMENTS FOR LARGE DATASET.

To illustrate the (theoretical) execution times on a large dataset, let us estimate the time to compute 10 and 100 iterations of the K-SVD algorithm on embeddings from the OpenCLIP dataset, which has 400 million samples. We will consider having a cluster with 8 nodes and 64 cores each, and compute in `Float32` precision.

As a datapoint, let's consider measurements from my 16 thread (8 core) Intel i7 mobile processor, which achieves 10 iterations on 800'000 samples in about 120 seconds. Using the setup above, we have about 32 times more compute resources and a 500 times larger problem, which yields an estimated runtime of $120\text{sec} \cdot \frac{500}{32} = 1875\text{sec} \approx 0.5\text{h}$, and similarly 5h for 100 iterations etc.

4 EXPERIMENTAL RESULTS.

4.1 SETUP.

We consider the `sbucaptions` dataset (Ordonez, Kulkarni, and Berg 2011) which contains one million captioned images. We process each image with the `openai/clip-vit-base-patch32` model, which produces embeddings of size 768. For the dictionary learning algorithm, we choose the number of dictionary elements as four times the embedding size, i.e. $n_{\text{dict}} = 4 \cdot 768 = 3072$. We further constrain each sample to be encoded by a maximum of 10 dictionary elements, i.e. $n_{\text{nnz}} = 10$ for the K-SVD algorithm.

For the caption processing we use the model `mixtral-8x7b` (Jiang et al. 2024) with the following system prompt:

> "This is an IQ test. The task is to find the underlying common themes in a series of image captions. Find the three to five main themes. Answer only in json format, with the fields 'themes' containing only the themes, and 'summary' containing the summary of each theme. Order the themes by how often they occur."

We then store the "summary" section for each concept. During evaluation time, we only consider the first of the concepts, i.e. the one the model found to occur the most often.

4.2 AUTOINTERPRETABILITY SCORE.

Fig. fig:autointerp-result depicts the results of the autointerpretability metric described in Sec. subsec:evaluating-explanations. For comparison, we also compute the metric for (i) the original descriptions, and (ii) for a "meaningless" text (lorem ipsum). As introduced in Sec. subsec:evaluating-explanations we find that the values fall into a relatively small range of approximately $[0.1, 0.4]$, which is consistent with previous findings. We also find that although our explanations do outperform the "meaningless" captions, the margin is not very large.

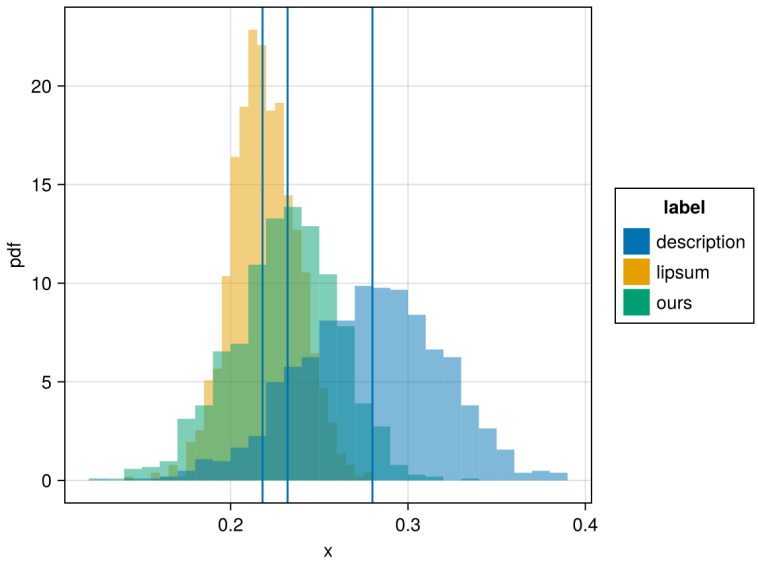


Figure 3: We plot the distribution of $d_{\text{cosine}}\left(CLIP(image), CLIP(text)\right)$ where $text$ is either the true image caption, our autointerpretability result, or a meaningless string, and $d_{\text{cosine}}(\cdot)$ is the cosine similarity (larger is better).

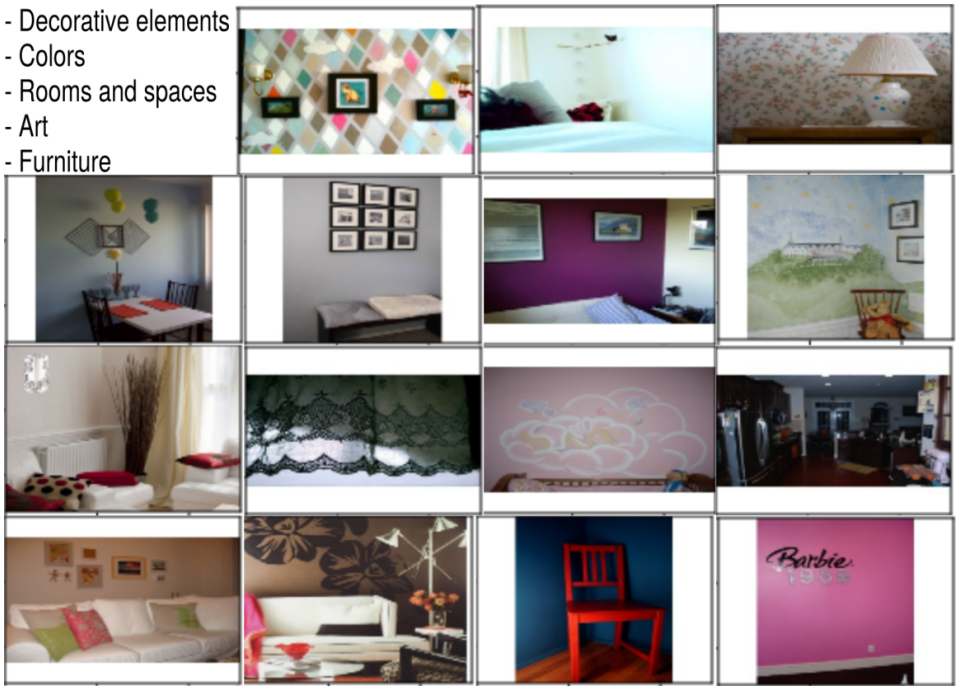


Figure 4: Vizualization of the first concept. Example images for which the concept is active, and the found themes (top left).

4.3 VISUAL INSPECTION.

Fig. fig:viz-1 depicts a subset of images for which the first concept is "active" on the left, and the list of autointerpretability results on the right. Further examples are provided in Sec. subsec:concept-visualization-appendix.

We can see that all the grouped images indeed share common themes. Notably, this grouping has been established using the concept assignment (matrix X) only, without having to refer to any captions. Further, we can see that the image captions do seem reasonable; however, they are also not very precise, listing several concepts at once.

4.4 DISCUSSION

The fact that Fig. fig:autointerp-result shows significant overlap between consine similarities of the (image, description) tuples and the (image, lorem ipsum) tuples indicates that better metrics to measure image captioning might be needed, or that the dataset might be flawed.

Regardless, our proposed disentanglement and captioning method does outperform the lorem ipsum "captions" although not by a wide margin.

In the "visual inspection", we can see that the assigned concepts seem mostly applicable, but are not very precise. This could mean:

- we need a larger set of dictionary elements
- the concepts are not disentangled well enough
- the setup doesn't force the algorithm to produce precise concept descriptions.

We will need to explore further to see if the "preciseness" can be measured and then improved.

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).

Bezanson, Jeff, Alan Edelman, Stefan Karpinski, and Viral B. Shah. 2017. "Julia: A Fresh Approach to Numerical Computing." *Siam Review*, no. 1 (January).

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*.

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."

Hessel, Jack, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. 2021. "CLIPScore: A Reference-free Evaluation Metric for Image Captioning." In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

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*.

Jiang, Albert Q., Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, et al. 2024. "Mixtral of Experts." January 8, 2024. `https://arxiv.org/abs/2401.04088`.

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.

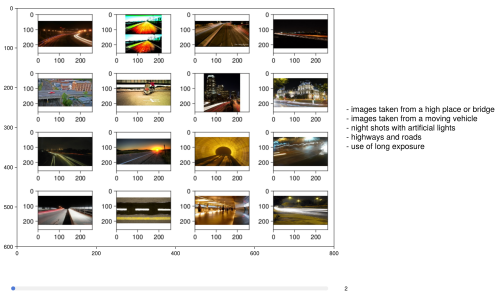
Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, et al. 2021. "Learning Transferable Visual Models From Natural Language Supervision." In *Proceedings of the 38th International Conference on Machine Learning*. PMLR.

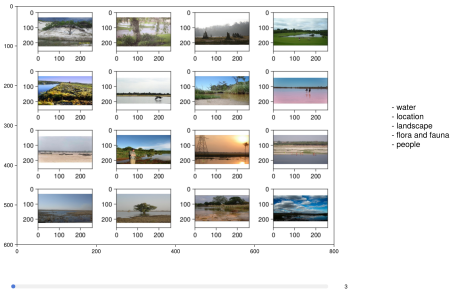
Steck, Harald, Chaitanya Ekanadham, and Nathan Kallus. 2024. "Is Cosine-Similarity of Embeddings Really About Similarity?" March 8, 2024. `https://arxiv.org/abs/2403.05440`.

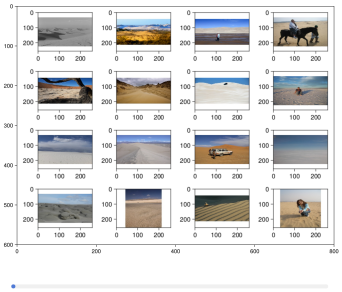
A Appendix

A.1 More concept visualizations.

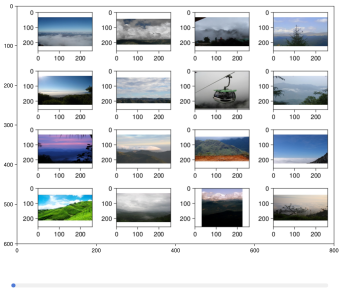
We present the first ten concept visualizations here. Further visualizations can be found at `https://github.com/RomeoV/vit_concept_visualization`.



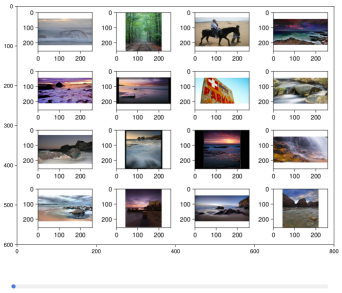




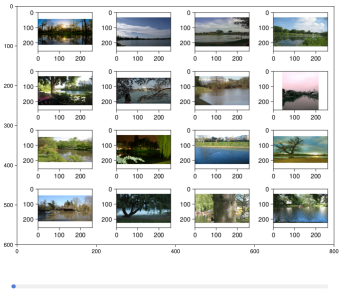
- Sand dunes
- Desert landscapes
- Beaches
- Mountains and hills



- Natural Landscapes
- Aerial Views
- Mountains and Hills
- Clouds
- Man-made Structures



- Natural Landscapes
- Artificial Structures
- Water
- Sky and Clouds
- Animals



- Water
- Nature and Landscapes
- Bodies of Water Surroundings
- Buildings and Structures

- Natural landscapes
- Man-made structures
- Activities and states
- Animal life

- Children
- Pose or Play
- Location
- Attire
- Animals

- field
- farm
- fence