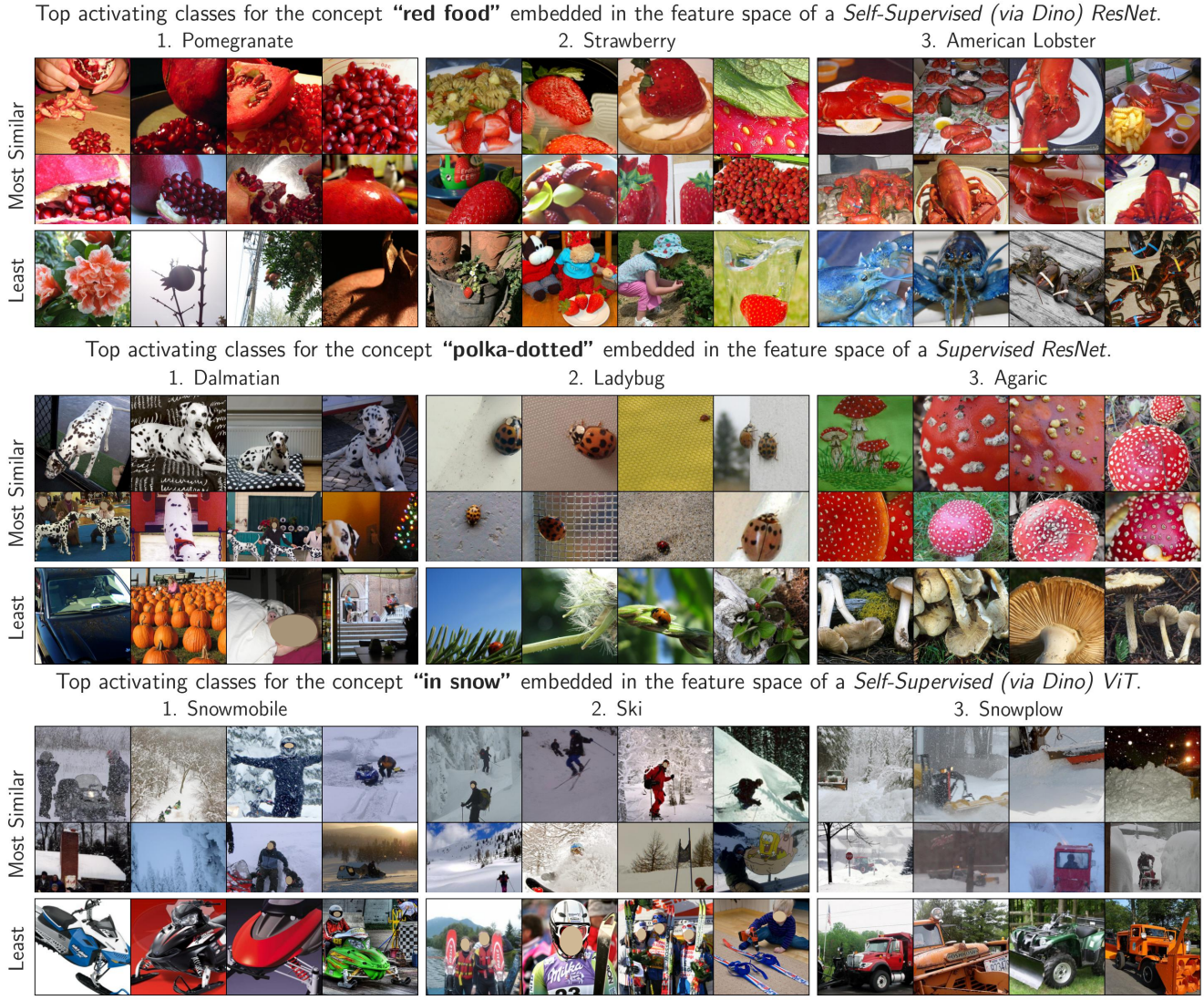


Figure 5. Additional qualitative validation of text-to-concept. Observe that the samples least similar to the concept vector for "in snow" break a common spurious correlation for their classes. Text-to-concept may then be used to identify challenging natural images within datasets, towards mitigating spurious correlation dependencies.

A. Prompts for Text-to-concept

When not otherwise specified, we use the default templates introduced in the original CLIP paper for ImageNet zero-shot classification. They are as follows: *'itap of a {}', 'a bad photo of the {}', 'a origami {}', 'a photo of the large {}', 'a {} in a video game', 'art of the {}', 'a photo of the small {}'*.

For Figure 1 and Figure 5, we append "in a tree" and "in snow" to the above templates, and also replace the '{}' with names for all ImageNet classes. The final concept vector is then an average of NUMBER OF TEMPLATES $\times$ NUMBER OF CLASES vectors. We do this because these correspond to contexts, which should be object agnostic. Similar results are obtained without refinement (i.e. replacing {} with 'object'). We note that embedding text to CLIP's space is very quick, only taking seconds to encode a batch of thousands of short phrases.

B. Zero-shot Classification

We now provide additional experimental details for Zero-shot classification via Text2Concept. We also include zero-shot accuracy for edge cases like recognizes characters or 'primitive' concepts, such as textures, colors, and shapes.

Dataset	Example Classes	Prompt	Citation
Coarse Grained Concepts In Distribution			
IN9	dog, bird, wheeled vehicle	a photo of {}	[34]
Living17	salamander, turtle, lizard	a photo of a {}	[29]
Nonliving26	bag, ball, boat	a photo of a {}	[29]
Entity13	garment, bird, reptile	a photo of a {}	[29]
Entity30	serpentes, passerine, saurian	a photo of a {}	[29]
Coarse Grained Concepts Out of Distribution			
CIFAR10	airplane, automobile, bird	a pixelated photo of a {}	[17]
STL10	airplane, bird, car	a photo of a {}	[6]
Fashion MNIST	T-shirt/top, Trouser, Pullover	a black and white photo of {}	[33]
CelebA Hair	brown hair, blonde hair	a headshot of a person with {}	[20]
Character Recognition			
SVHN	zero, one, two	a photo of the digit "{}" on a building	[24]
MNIST	zero, one, two	a photo of the digit "{}"	[18]
Primitive Concepts			
Textures	banded, blotchy, braided	a photo of something with {} texture	[5]
Color	black, blue, brown	a swatch of the color {}	-
Shape	circle, octagon, square	a diagram of the shape {}	[16]

Table 1. List of datasets studied in Zero-Shot classification experiments (Section 3.2), along with example classes and the specific prompt used. Note that we use an internal simple dataset for *Color*.

B.1. Experimental Details

We carry out zero-shot experiments over many datasets. We use slightly different prompts for each task, though we stress that we did not optimize prompt engineering to obtain better results. All evaluated models use the same prompts. Table 1 shows details for prompts used, as well as example classes for each dataset, to give an idea as to what kind of text is used to generate concept vectors. We refer readers to the original sources for more details on the datasets studied.

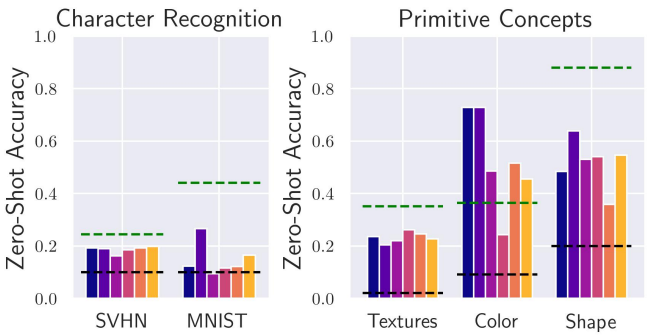


Figure 6. Edge cases for zero-shot classification. (**Left**) Models struggle with OCR. (**Right**) Models can recognize some primitive concepts by name. Same legend as figure 2.

For color recognition, we construct a simple dataset that consists of one sample per the following classes: *black, blue, brown, gray, green, orange, pink, purple, red, white, yellow*. The sample in each class is a monocolor patch, with every pixel set to have the color given by the class name. Also, for shape recognition, we use a subset of the shapes in the original dataset. Specifically, we include the following shapes: *circle, octagon, square, star, triangle*.

B.2. Edge Cases

To stress test Text2Concept, we consider tasks that require models to recognize characters (specifically digits) or primitive concepts, like textures, colors, and shapes. We observe most models only marginally surpass random accuracy for character recognition tasks. Oddly, the adversarially trained ResNet is roughly twice as good as other models in zero-shot MNIST classification, though it still performs far worse than the baseline CLIP model, which also struggles. This suggests models simply may not have any notion as to what distinguishes digits from one another, which is not surprising given that it would not be very useful for understanding ImageNet images. On the other hand, models achieve

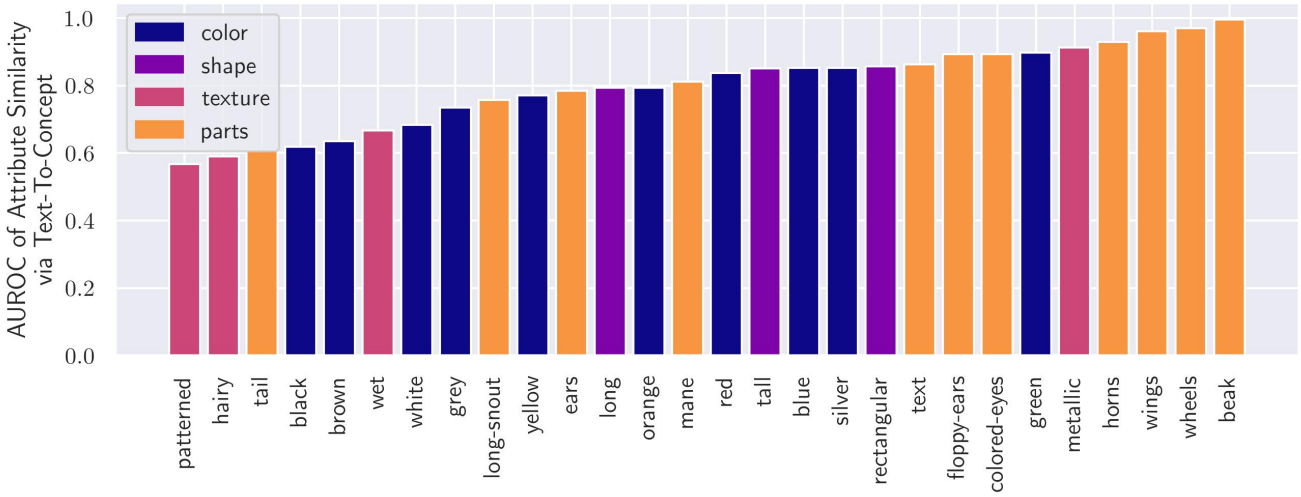


Figure 7. Quality of using similarity to text-to-concept vectors for predicting RIVAL10 attributes. AUROC shown per attribute. Attributes corresponding to parts are predicting more reliably. Over 70% of attributes achieve an AUROC of at least 0.75.

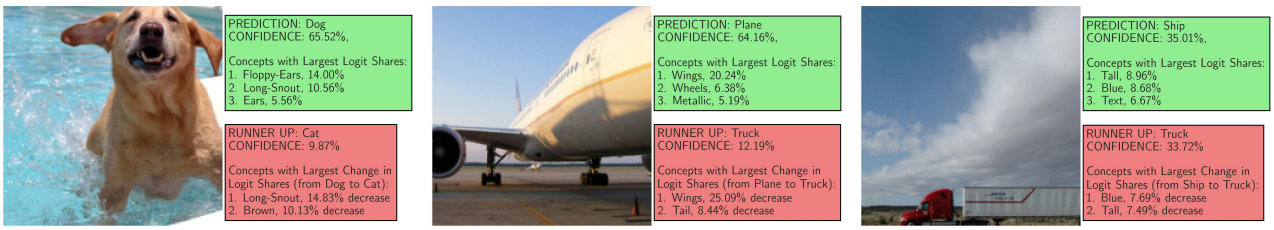


Figure 8. Extra examples of inference using the concept-bottleneck model (enabling direct measurement of each concept's contribution to a class logit, as shown) built atop a fixed vision encoder and text-to-concept. We include a misclassification in the rightmost panel.

D. Limitations

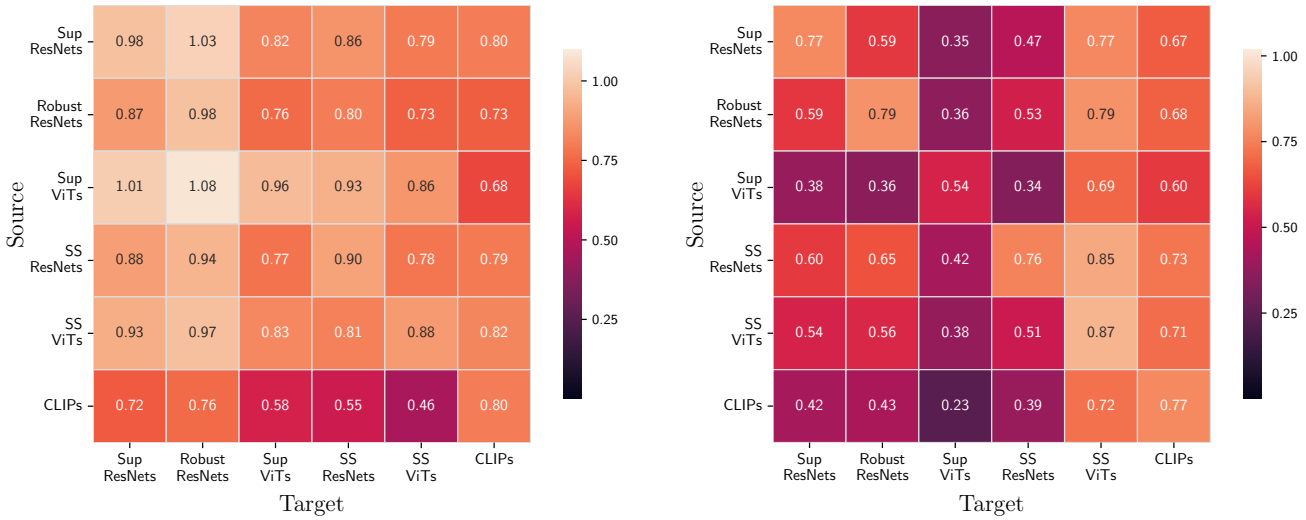
We note that the concept vectors we find are not always perfect. However, more refined concept vectors can be obtained with (1) better prompt engineering and (2) extraction of exemplar data to that concept. Note that Text2Concept enables step 2 without needing to collect new data. That is, we can better organize our data by sorting it with respect to similarity to certain Text2Concept vectors. Then, the images with highest and least similarity can serve as good positive and negative examples for the concept. Moreover, we can select data in a class balanced way, so to disentangle a concept from a class that it is correlated with. As an example, consider the concept "in snow". Arbitrarily selecting images with highest and lowest similarity likely will lead to un-informative negative examples. However, if we select examples within classes that are have high average similarity to the concept, then we can obtain more challenging negative examples that better distill the essence of the concept of interest. See Figure 5: simply reorganizing data based on similarity to the corresponding Text2Concept vectors leads to easy access of informative exemplars within the data one has already collected, towards more refined CAVs.

These images can be further used to find a better concept vector, either by simply taking average of their encoded representations for positive examples, or by training a linear classifier in feature space to distinguish the positive and negative examples, as was originally performed in [14].

E. Alignment

We now more formally describe our procedure for linearly mapping a source feature space to a target space (e.g. CLIP's latent space). We also provide experimental results over an expansive set of diverse model pairs, demonstrating the surprising effectiveness of the simple linear mapping.

We use $\mathcal{X}$ to denote the set of all possible input images. Let $D_{\text{train}}, D_{\text{test}} \subset \mathcal{X}$ denote the training and test datasets. We define a *vision encoder* as a model f that maps images $x \in \mathcal{X}$ to vectors $f(x) \in \mathbb{R}^d$. Given two vision encoders f_s, f_t,



(a) The average of retained accuracy.

(b) The average of R^2.

Figure 9. **(Left)** shows the average of retained accuracy. More precisely, the value in row r and column c is the average of retained accuracy when doing alignment from representation space of model s to that of model t where s is a model in group r and t is a model of group c. **(Right)** shows the average of R^2, i.e., same as above, value in row r and column c is the average of R^2 in linear alignment from models in group r to models of group c. Note that "Sup" stands for Supervised while "SS" stands for Self-Supervised training procedure. Note that all models are pretrained on ImagenNet-1K except CLIPs. More details on models used here can be found in Section E.2.

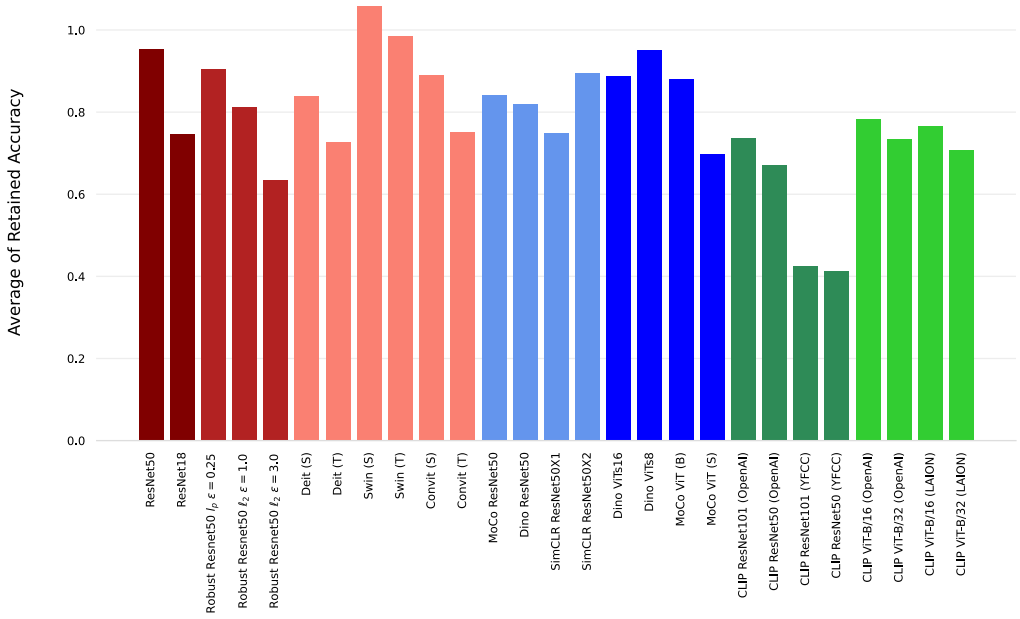


Figure 10. For each model s, average of retained accuracy when doing alignment from model s to all other models is reported.

representation space alignment of model f_s to model f_t is the task of learning a mapping $h : f_s(\mathcal{X}) \to f_t(\mathcal{X})$. We restrict h to the class of affine transformations, i.e., $h_{W,b}(z) := W^T z + b$.

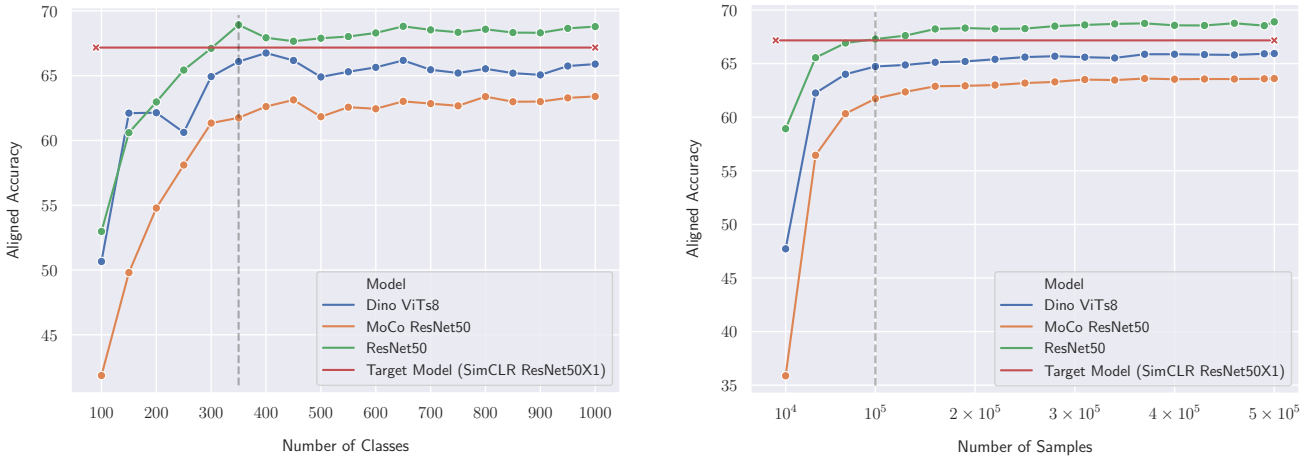


Figure 11. **(Left)** shows the aligned accuracy when linear transformation is only optimized on images with particular labels. We randomly select labels and increase the number of labels(classes) to see how retained accuracy changes. while **(Right)** shows the aligned accuracy when linear alignment is solved on a random subset of images. Alignment is done from three different models to SimCLR ResNet50X1. We observe that all training images are not necessary to have a reliable alignment. In other words, aligned accuracy can reach to its maximum by only considering small portion of images or classes.

To maximally retain the original semantics of representation spaces, we design the following optimization problem

$$W, b = \arg\min_{W,b} \frac{1}{|D_{\text{train}}|} \sum_{x \in D_{\text{train}}} \|W^T f_s(x) + b - f_t(x)\|_2^2. \tag{1}$$

The above optimization can be viewed as multiple linear regression problems; thus we evaluate the linear alignment on D_{test} by considering the quality of the solution on those linear regression problems. We use *Coefficient of Determination*, i.e., R^2 which is the proportion of the variation in the dependent variables that is predictable from the independent variables. Furthermore, we note that for the vision encoder f_s, there usually exists a *classification head* $g_s : \mathbb{R}^d \to \mathcal{C}$ that classifies a representation in the space of model f_s. Indeed, the predicted label for input x is $g_s(f_s(x))$. Note that $\mathcal{C}$ denotes the set of labels, e.g., ImageNet classes. We define *aligned accuracy* as the accuracy of classification on D_{test} when we use f_s as the vision encoder, then do the linear transformation to obtain the corresponding representation in space of f_t, and finally, use g_t for classification. If alignment works well, aligned accuracy should be admissible and comparable to the accuracy of model f_t when no alignment is used. Then, we define *retained accuracy* as the ratio of aligned accuracy to the accuracy of model f_t without any alignment. Note that we use ImageNet-1K train and test datasets as D_{train} and D_{test} in linear alignment.

Interestingly, we observe that simple linear alignment works well in terms of both R^2 and aligned accuracy across various models. Figure 9 shows the aligned accuracy and R^2 between diverse pairs of models. We find that **various models are highly alignable to CLIP models**. This is surprising as CLIP models are trained on other datasets than ImageNet and their training procedure involves vision/text supervision which is drastically different from other models. High-quality alignment to CLIP representation space enables models to adopt a wide variety of CLIP models capabilities, which we analyze in this work. On the other hand, we observe that retained accuracy when aligning CLIP models to other models is not high. This is mainly due to the fact that CLIP models encode images and texts in relatively low-dimensional spaces and in linear regression, approximating dependent variables becomes harder as the number of independent variables decreases. Indeed, linear alignment works worse when we align from a representation space with lower dimensionality to a representation space with higher dimensionality.

E.1. Optimizing the Linear Transformation

With a proper set of hyperparameters, around 6 epochs are enough to converge to the optimal solution. However, re-scaling representation spaces of models so that the variance of elements in the space becomes constant, is crucial. This is due to the fact that some models embed inputs into very low variance spaces, which degrades the performance of linear alignment due to precision in computations.

Additionally, we take into account the optimization problem given in (1) and consider the effect of the number of images that we involve in optimizing (1). We observe that using only a random subset of the training set of ImageNet is sufficient to find W and b, as seen in Figure 11, $1/5$ of ImageNet training samples is roughly enough to retrieve the target model accuracy. if we use only images of some particular classes to optimize (1), we can retrieve the target accuracy by just using around $1/3$ of ImageNet classes.

The specific hyperparameters we use to solve alignment are as follows: we use SGD optimizer and learning rate scheduler (implemented in Torch [26]) with following hyperparameters:

```
optimizer = optim.SGD(lr=0.01, momentum=0.9, weight_decay=5e-4)
scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(T_max=200)
```

We run optimization for 6 epochs. Note that before optimizing we re-scale representation spaces of models such that variance of elements in matrix $\left(f_s(x_1^{\text{train}}), f_s(x_2^{\text{train}}), ..., f_s(x_N^{\text{train}})\right)$ becomes 4.5.

E.2. Models in our Study

In this paper we have considered several different models in different categories [27], [4], [3], [11], [28], [27]. All these models except CLIP models are trained on ImageNet-1K [8]. For almost all of these models, pretraiend weights are obtained from timm library [32] and [12]. Our models are categorized in following groups.

- **Supervised ResNets** include ResNet50, and ResNet18.
- **Robust ResNets** include Robust Resnet50 $\ell_2, \epsilon = 0.25$, Robust Resnet50 $\ell_2, \epsilon = 1.0$, and Robust Resnet50 $\ell_2, \epsilon = 3.0$.
- **Supervised Vision Transormers** include Swin, Deit, and Convit models.
  - Swin with patch size of 4 and window size of 7 includes Swin Small (S) and Swin Tiny (T).
  - Deit with patch size of 16 includes Deit Small (S) and Deit Tiny (T).
  - Convit includes Convit Small (S) and Convit Tiny (T).
- **Self-Supervised ResNets** include MoCo ResNet50, Dino ResNet50, SimCLR ResNet50X1, and SimCLR ResNet50X2.
- **Self-Supervised Vision Transformers** include MoCo ViT base (B), MoCo ViT small (S), Dino ViTs 16, and Dino ViTs 8, .
- **CLIP** include
  - CLIP ResNet101, CLIP ResNet50, CLIP ViT-B/16, and CLIP ViT-B/32 trained on OpenAI dataset.
  - CLIP ResNet101 and CLIP ResNet50 trained on YFCC [31].
  - CLIP CLIP ViT-B/16 and CLIP CLIP ViT-B/32 trained on LAION [30].

F. Discussion and Future Work

While our proposed method is very simple, we believe it can open the door to many variants that are better suited for specific aims. A central takeaway (and surprising observation, on which our method is based) is that the representation spaces of diverse models are surprisingly similar, to where even the simplest manner of mapping them (i.e. via linear regression) can be effective. Other optimizations, such as Nystrom kernel regression or maximizing similarity measures like CKA or HSIC, may be promising. More simply, we can use Text2Concept vectors to organize data, essentially mining for positive and negative examples within an existing dataset. Then, a refined CAV can be obtained in the traditional fashion (i.e. training a binary classifier linearly separating positive and negative examples of a concept in feature space). Lastly, it may be possible to obtain aligners in closed form, given the simplicity of our optimization task.

We mention these approaches as potential future directions for improving Text2Concept. Also, there are numerous interpretability benefits beyond what we present that may be worth pursuing. The ability to generate new CAVs for free should open the door to many new forms of interpretability. Finally, we situate our work in the broader moment of what we see as a rise of multi-modal methods in AI. Having developed very effective specialized models, bridging modalities may have massive impact in better utilizing the models we already have. Our work focuses on building these bridges in a highly efficient (w.r.t. time and samples) and extensible fashion, in contrast to other methods that have much greater computational and data requirements [35] or are more task-specific in their formulation [21].