Decomposing and Interpreting Image Representations via Text in ViTs Beyond CLIP

Sriram Balasubramanian [1] **Samyadeep Basu** [1] **Soheil Feizi** [1]

Abstract

Recent works have explored how individual components of the CLIP-ViT model contribute to the final representation by leveraging the shared image-text representation space of CLIP. Components like attention heads and MLPs have been found to capture distinct image features such as shape, color, or texture. However, understanding the role of these components in arbitrary vision transformers (ViTs) is challenging. Thus, we introduce a general framework which can identify the roles of various components in ViTs beyond CLIP. Specifically, we (a) automate the decomposition of the final representation into contributions from different model components, and (b) linearly map these contributions to CLIP space to interpret them via text. We also introduce a novel scoring function to rank components by their importance with respect to specific features. Applying our framework to various ViTs (e.g. DeiT, DINO, DINOv2, Swin, MaxViT), we gain insights into the roles of different components concerning particular image features.These insights facilitate applications such as image retrieval, visualizing token importance heatmaps, and mitigating spurious correlations.

1. Introduction

Vision transformers and their variants (Dosovitskiy et al., 2021; Oquab et al., 2023; Caron et al., 2021; Tu et al., 2022; Liu et al., 2021; Touvron et al., 2021) have emerged as powerful image encoders, becoming the preferred architecture for modern image foundation models. However, the mechanisms by which these models transform images into representation vectors remain poorly understood. Recently, Gandelsman et al. (2024) made significant progress on this question for CLIP-ViT models with two key insights: (i) They demonstrated that the residual connections and attention mechanisms of CLIP-ViT enable the model output to be mathematically represented as a sum of vectors over layers, attention heads, and tokens, along with contributions from MLPs and the CLS token. Each vector corresponds to the contribution of a specific token attended to by a particular attention head in a specific layer. (ii) These contribution vectors exist within the same shared image-text representation space, allowing the CLIP text encoder to interpret each vector individually via text.

Extending this approach to other transformer-based image encoders presents several challenges. Popular models like DeiT (Touvron et al., 2021), DINO-ViT (Caron et al., 2021; Oquab et al., 2023), and Swin (Liu et al., 2021) lack a corresponding text encoder to interpret the component contributions. Additionally, extracting the contribution vectors corresponding to these components is not straightforward, as they are often not explicitly computed during the forward pass of the model. Other complications include diverse attention mechanisms such as grid attention, block attention (in MaxViT), and windowed/shifted windowed attention (in Swin), as well as various linear transformations like pooling, downsampling, and patch merging applied to the residual streams between attention blocks. These differences necessitate a fresh mathematical analysis for each model architecture, followed by careful application of necessary transformations to the intermediate output of each component to determine its contribution to the final representation. To address these challenges, we propose our framework to identify roles of components in general ViTs.

First, we *automate the decomposition of the representation* by leveraging the computational graph created during the forward pass. This results in a drop-in function, RepDecompose, that can decompose any representation into contributions vectors from model components simply by calling it on the representation. Since this method operates on the computational graph, it is agnostic to the specifics of the model implementation and thus applicable to a variety of model architectures.

Secondly, we introduce an algorithm, CompAlign, to *map each component contribution vector to the image representation space of a CLIP model*. We train these linear maps with

[1]Department of Computer Science, University of Maryland, College Park. Correspondence to: Sriram Balasubramanian <sriramb@cs.umd.edu>.

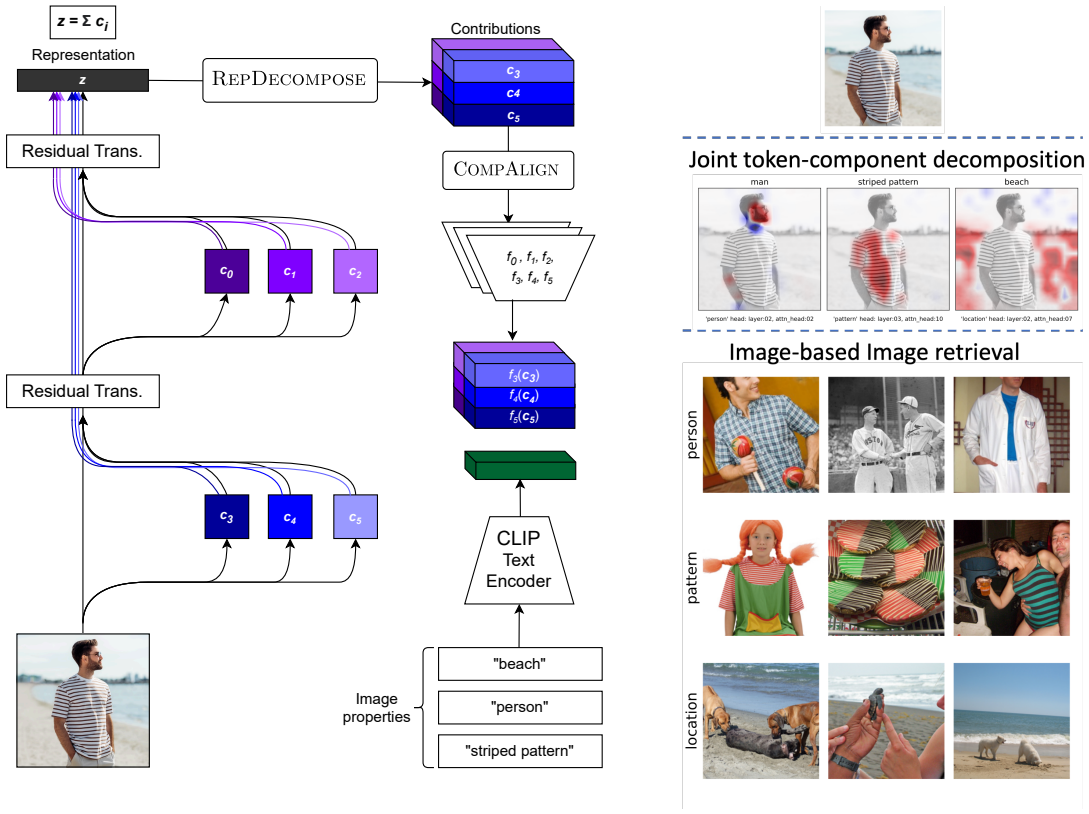

Figure 1: **(Left) Workflow:** The first step (REPDECOMPOSE) is to decompose a representation $\boldsymbol{z}$ into contributions from its model components $\boldsymbol{c}_i$ after being transformed by residual transformations like LayerNorm, linear projections, resampling, patch merging and so on. The second step (COMPALIGN) aligns each contribution to CLIP space using a set of linear maps $f_0, f_1, \ldots, f_n$ on the corresponding contributions $\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$. We can then interpret these aligned contributions using the CLIP text encoder. **(Right) Applications of our method:** (a) Visualizing contributions of each token through a specific component using a joint token-component decomposition (b) Retrieving images that are close matches of the reference image (on top) with respect to a given image feature like pattern, person, or location

regularizations so that these maps preserve the roles of the individual components while also aligning the model's image representation with CLIP's image representation. This allows us to map each contribution vector from any component to CLIP space, where they can be interpreted through text using a CLIP text encoder.

Thirdly, we observe that there is often *no straightforward one-to-one mapping* between model components and common image features such as shape, pattern, color, and texture. Sometimes, a single component may encode multiple features, while multiple components may be required to fully encode a single feature. To address this, we propose a *scoring function* that assigns an importance score to each component-feature pair. This allows us to rank components based on their importance for a given feature, and rank features based on their importance for a given component.

Using this ranking, we proceed to analyze diverse vision transformers such as DeiT, DINO, Swin, and MaxViT, in addition to CLIP, in terms of their components and the image features that they are responsible for encoding. We consistently find that many components in these models encode the same feature, particularly in ImageNet pre-trained models. Additionally, individual components in larger models MaxVit and Swin do not respond to any image feature strongly, but can encode them effectively in combination with other components. This diffuse and flexible nature of feature representation underscores the need for interpreting them using a continuous scoring and ranking method as opposed to labelling each component with a well-defined role. We are thus able to perform tasks such as image retrieval, visualizing token contributions, and spurious correlation mitigation by carefully selecting or ablating specific components based on their scores for a given property.

Algorithm 1 REPDECOMPOSE

Input: $\boldsymbol{z}$, the final representation output by the model and the final node in the computational graph. $\boldsymbol{z}.f$ is the function that outputs node $\boldsymbol{z}$

Output: A tree $\boldsymbol{t}$ consisting of component contributions $\boldsymbol{c}$, such that components $\sum_{\boldsymbol{c}\in\boldsymbol{t}} \boldsymbol{c} = \boldsymbol{z}$. The structure of $\boldsymbol{t}$ is a nested list where each list represents a level in the tree

function REPDECOMPOSE($\boldsymbol{z}$)
- **if** is_nonlinear($\boldsymbol{z}.f$) **then**
  - **return** $[z]$
- **else if** is_unary($\boldsymbol{z}.f$) **then** ▷ *Function is unary linear*
  - $\boldsymbol{z}_0 \leftarrow \boldsymbol{z}$.parents()
  - $\boldsymbol{t}_0 \leftarrow$ REPDECOMPOSE($\boldsymbol{z}_0$)
  - **if** is_reduction($\boldsymbol{z}.f$) **then**
    - $\boldsymbol{t}_{0,u} \leftarrow$ unbind($\boldsymbol{t}_0$) ▷ *Unbinds each $\boldsymbol{c} \in \boldsymbol{t}$ along the reduction dimension*
    - $f_d \leftarrow$ decomp($\boldsymbol{z}.f$) ▷ *Returns f_d such that $\sum_{\boldsymbol{c}\in\boldsymbol{t}_{0,u}} f_d(\boldsymbol{c}) = \boldsymbol{z}.f(\boldsymbol{z}_0)$*
    - **return** map(f_d, $\boldsymbol{t}_{0,u}$) ▷ *Maps each $\boldsymbol{c} \in \boldsymbol{t}_{0,u}$ to $f_d(\boldsymbol{c})$*
  - **else**
    - **return** map($\boldsymbol{z}.f$, $\boldsymbol{t}^0$) ▷ *Maps each element $\boldsymbol{c} \in \boldsymbol{t}$ to $\boldsymbol{z}.f(\boldsymbol{c})$*
- **else** ▷ *$\boldsymbol{z}.f$ is binary*
  - $\boldsymbol{z}_0, \boldsymbol{z}_1 \leftarrow \boldsymbol{z}$.parents() ▷ *Get the parents of $\boldsymbol{z}$ in the graph (inputs to $\boldsymbol{z}$.function)*
  - $\boldsymbol{t}_0, \boldsymbol{t}_1 \leftarrow$ REPDECOMPOSE($\boldsymbol{z}_0$), REPDECOMPOSE($\boldsymbol{z}_1$)
  - $f_{d,0}, f_{d,1} \leftarrow$ decomp_binary($\boldsymbol{z}.f$) ▷ *Returns $f_{d,0}, f_{d,1}$ such that:*
  - **return** [map($f_{d,0}$, $\boldsymbol{t}_0$), map($f_{d,1}$, $\boldsymbol{t}_1$)] ▷ $\sum_{\boldsymbol{c}\in\boldsymbol{t}_0} f_{d,0}(\boldsymbol{c}) + \sum_{\boldsymbol{c}\in\boldsymbol{t}_1} f_{d,1}(\boldsymbol{c}) = \boldsymbol{z}.f(\boldsymbol{z}_0, \boldsymbol{z}_1)$

2. Decomposing the Final Representation

Recently, Gandelsman et al. (2024) decomposed $\boldsymbol{z}_{\text{CLS, fin}}$, the final [CLS] representation of the CLIP's image encoder as a sum over the contributions from its attention heads, layers and token positions, as well as contributions from the MLPs. Thus, this representation can be decomposed as: $\boldsymbol{z}_{\text{CLS, fin}} = \boldsymbol{z}_{\text{CLS, init}} + \sum_{l=1}^{L} c_{l,\text{MLP}} + \sum_{l=1}^{L}\sum_{h=1}^{H}\sum_{t=1}^{N} c_{l,h,t}$, where L, H, N correspond to the number of layers, number of attention heads and number of tokens. Here, $c_{l,h,t}$ denotes the contribution of token t though attention head h in layer l, while $c_{l,\text{MLP}}$ denotes the contribution from the MLP in layer l. Due to this linear decomposition, different dimensions can be reduced by summing over them to identify the contributions of tokens or attention heads to the final representation. While this decomposition is relatively simple for vanilla ViTs, it cannot be directly used for general ViT architectures due to use of self-attention variants such as window attention, grid attention, or block attention, combined with operations such as pooling or patch merging on the residual stream. The final representation may also not just be a single $\boldsymbol{z}_{\text{CLS, fin}}$ but $\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{z}_{i,\text{fin}}$ or even $\frac{1}{L}\sum_{i=1}^{L} \boldsymbol{z}_{\text{CLS},i}$.

We thus seek a general algorithm which can automatically decompose the representation for general ViTs. This can be done via a recursive traversal of the computation graph. Suppose the final representation $\boldsymbol{z}$ can be decomposed into component contributions $\boldsymbol{c}_{i,t}$ such that $\boldsymbol{z} = \sum_{i,t} \boldsymbol{c}_{i,t}$. Here each $\boldsymbol{c}_{i,t}$ corresponds to the contribution of a particular token t through some model component i. For convenience, let $\boldsymbol{c}_i = \sum_t \boldsymbol{c}_{i,t}$. Then, if given access to the computational graph of $\boldsymbol{z}$, we can identify potential linear components $\boldsymbol{c}_{i,t}$ by recursively traversing the graph starting from the node which outputs $\boldsymbol{z}$ in reverse order till we hit a non-linear node. The key insight here is that the output of any node which performs a linear reduction (defined as a linear operation which results in a reduction in the number of dimensions) is equivalent to a sum of individual tensors of the same dimension as the output. These tensors can be collected and transformed appropriately during the graph traversal to obtain a list of tensors $\boldsymbol{c}_{i,t}$, each members of the same vector space as the representation $\boldsymbol{z}$. This kind of linear decomposition is possible due to the overwhelmingly linear nature of transformers. The high-level logic of REPDECOMPOSE is detailed in Algorithm 1. In practice, the number of components quickly explodes as there are a very large number of potential component divisions for a given model. Thus, we restrict it to only the attention heads and MLPs with no finer divisions. We also constrain REPDECOMPOSE to return only the *direct* contributions of these components to the output. In principle, REPDECOMPOSE could return higher order terms such as $\boldsymbol{c}_{j,i}$ which is the contribution of model component i via the downstream component j. We defer analysis of these higher order components for future work.

Embedding Source	ImageNet pretrained	One map only	CompAlign ($\lambda = 0$)	CompAlign
TextSpan's top 10 descriptions of a random component	wardrobe	gyromitra	bookcase	filing cabinet
	medicine cabinet	home theater	snorkel	snorkel
	window shade	drumstick	red wolf	bakery
	desk	Samoyed	barbershop	bathtub
	barbershop	muzzle	microwave oven	dining table
	refrigerator	bookstore	bassinet	red wolf
	library	dining table	disc brake	gyromitra
	shoji screen	medicine cabinet	dining table	shoji screen
	bathtub	park bench	sink	Norwich Terrier
	dining table	tusker	window screen	bookstore
Match rate	-	0.08	0.155	0.185
Cosine Distance	-	0.23	0.18	0.17

Table 1: Comparison of different methods to map the representation space of ImageNet-1k pre-trained DeiT-B/16 to CLIP image representation space. The green colored texts are exact matches with the top-10 descriptions obtained from the imagenet pretrained embeddings, while the orange colored texts are approximate matches. The match rate is the average fraction of exact matches across all components, while cosine distance is the average cosine distance between the CLIP representations and the transformed model representations on ImageNet

3. Aligning the component representations to CLIP space

Having decomposed the representation into contributions from relevant model components, we now aim to interpret these contributions through text using CLIP by mapping them to CLIP space. Formally, given that we have a set of vectors $\{\boldsymbol{c}_i\}_{i=1}^N$ such that $\sum_i^N \boldsymbol{c}_i = \boldsymbol{z}$, the final representation of model, we require a set of linear maps f_i such that the sum of $\sum_i f_i(\boldsymbol{c}_i) = \boldsymbol{z}_{\text{CLIP}}$, the final representation of the CLIP model. Once we have these CLIP aligned vectors, we can interpret them via text using CLIP text encoders.

However, from an interpretability standpoint, a few additional constraints on the linear maps are desirable. Consider a component contribution $\boldsymbol{c}_i$ and two directions $\boldsymbol{u}, \boldsymbol{v}$ belonging to the same vector space as $\boldsymbol{c}_i$ which represent two distinct features, say shape and texture. Let us further assume that the component's dominant role is to identify shape, and thus the variance of the projection of $\boldsymbol{c}_i$ along $\boldsymbol{u}$ is higher than that of $\boldsymbol{v}$. We want this relative importance of features to be maintained in $f_i(\boldsymbol{c}_i)$. Additionally, we also want any two linear maps f_i and f_j to not change the relative norms of features in components $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$. We can express these conditions formally as follows:

1. **Intra-component norm rank ordering**: For any two vectors $\boldsymbol{u}, \boldsymbol{v}$ and a linear map f_i such that $\|\boldsymbol{u}\| \leq \|\boldsymbol{v}\|$, we have $\|f_i(\boldsymbol{u})\| \leq \|f_i(\boldsymbol{v})\|$
2. **Inter-component norm rank ordering**: For any two vectors $\boldsymbol{u}, \boldsymbol{v}$ and linear maps f_i, f_j such that $\|\boldsymbol{u}\| \leq \|\boldsymbol{v}\|$, we have $\|f_i(\boldsymbol{u})\| \leq \|f_j(\boldsymbol{v})\|$

Theorem 1. *Both of the above conditions together imply that all linear maps f_i must be a scalar multiple of an orthogonal transformation, that is for all i, $f_i^T f_i = kI$ for some constant k. Here, I is the identity transformation.*

The proof is deferred to appendix C. We can now formulate a novel alignment method, CompAlign, to map contributions of model components to CLIP space. CompAlign minimizes a loss function over $\{f_i\}_{i=1}^N$ to obtain a good alignment between model representation space and CLIP space:

$$L(\{f_i\}_{i=1}^N) = \mathbb{E}_{\{\boldsymbol{c}_i\}_{i=1}^N, \boldsymbol{z}_{\text{CLIP}}} \left[1 - \cos\left(\sum_i f_i(\boldsymbol{c}_i), \boldsymbol{z}_{\text{CLIP}} \right) \right] + \lambda \sum_i \|f_i^T f_i - I\|_F$$

The first term of the objective is the *alignment loss*, which is the average cosine distance between the CLIP representation $\boldsymbol{z}_{\text{CLIP}}$ and the transformed model representation $\sum_i f_i(\boldsymbol{c}_i)$. It quantifies the directional discrepancy between the two vectors. The second term is the *orthogonality regularizer* which imposes a penalty if the linear maps f_i are not orthogonal, ensuring that the f_i adhere closely to the specified conditions. We can now train f_i using the above loss function on ImageNet-1k. The training is label-free and can be done even over unlabeled datasets. We obtain $\boldsymbol{z}_{\text{CLIP}}$ from the CLIP image encoder and $\{\boldsymbol{c}_i\}_{i=1}^N$ from running RepDecompose on the final representation of the model.

Ablation study: We now conduct an ablation study on CompAlign. The first naive alignment method uses the same linear map f for all f_i without any constraints (Moay-

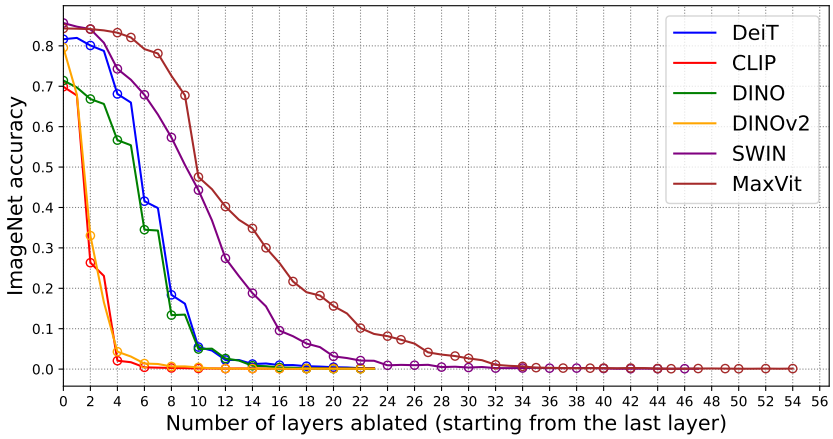

Figure 2: Ablation results for various different image encoders. The top-1 ImageNet accuracy is plotted as the layers of the model are increasingly ablated away, starting from the last layer up till the first layer. The circles on the plot represent the endpoints of blocks, the definition of which varies across model architectures. For the vanilla ViT variants, a block is an attention MLP pair, while for SWIN, it is a pair of windowed/shifted windowed attention and an MLP. For MaxVit, this might either be a grid/block attention-MLP pair, or an MBConv block.

eri et al., 2023). The second method is a version of COMPALIGN with $\lambda = 0$, where all f_i are different but not trained with the orthogonality regularizer. To compare these methods, we first get a "ground truth" description for each model component by using the TEXTSPAN (Gandelsman et al., 2024) algorithm on the class embedding vectors from the ImageNet pre-trained head. This yields descriptions of each component in terms of the top 10 most dominant ImageNet classes. We then use COMPALIGN and the two baselines to map the representations to CLIP space, and apply TEXTSPAN on CLIP embedded ImageNet class vectors to label each model component. We can then compare the descriptions this yields with the "ground truth" text description for each head. The results, shown in Tab. 1, indicate that COMPALIGN's TEXTSPAN descriptions have the most matches to the ImageNet pre-trained descriptions, followed by COMPALIGN with $\lambda = 0$ and the naive single map method. This trend is similar in the average cosine distance between the CLIP representations and the transformed model representations.

4. Component ablation

To identify the most relevant model layers for downstream tasks, we progressively ablate them and measure the drop in ImageNet classification accuracy. Ablation involves setting a layer's contribution to its mean value over the dataset. We use the following models from Huggingface's `timm` (Wightman, 2019) repository: (i) DeiT (ViT-B-16) (Touvron et al., 2021), (ii) DINO (ViT-B-16) (Caron et al., 2021), (iii) DINOv2 (ViT-B-14) (Oquab et al., 2023), (iv) Swin Base (patch size = 4, window size = 7) (Liu et al., 2021), (v) MaxViT Small (Tu et al., 2022), along with (vi) CLIP (ViT-B-16) (Cherti et al., 2023) from `open_clip` (Ilharco et al., 2021). DeiT, Swin, and MaxViT are pretrained on ImageNet with a supervised classification loss, DINO on ImageNet with a self-supervised loss, DINOv2 on LVD-142M with a self-supervised loss, while CLIP is pretrained on a LAION-2B subset with contrastive loss.

In Fig. 2, we see that for models not trained on ImageNet (CLIP and DINOv2), removing the last few layers quickly drops the accuracy to zero. In contrast, models trained on ImageNet experience a more gradual decline in accuracy, reaching zero only after more than half the layers are ablated. This trend is consistent across both self-supervised (DINO) and classification-supervised (DeiT, SWIN, MaxViT) models. This suggests that ImageNet-trained models encode useful features redundantly across layers for the classification task. Additionally, larger models with more layers, such as MaxVit, show significantly more redundancy, with minimal accuracy impact from ablating the last four layers. Conversely, the first few layers in all models contribute little to the output. Therefore, our analysis in the subsequent sections is focused on the last few layers of each model.

5. Feature-based component analysis

We now analyze the final representation in terms finer components like attention heads and MLPs, focusing on the last few significant layers. We limit decomposition to 10 layers for DeiT, DINO, and DINOv2, but 12 layers for SWIN and 20 layers for MaxVit due to their greater depth and redundancy across components. We accumulate contributions from the remaining components in a single vector $\boldsymbol{c}_{\text{init}}$, expressing $\boldsymbol{z}$ as $\boldsymbol{c}_{\text{init}} + \sum_i^N \boldsymbol{c}_i$, where N is the total number of components excluding $\boldsymbol{c}_{\text{init}}$. Here, $N = 65$ for DeiT, DINO,

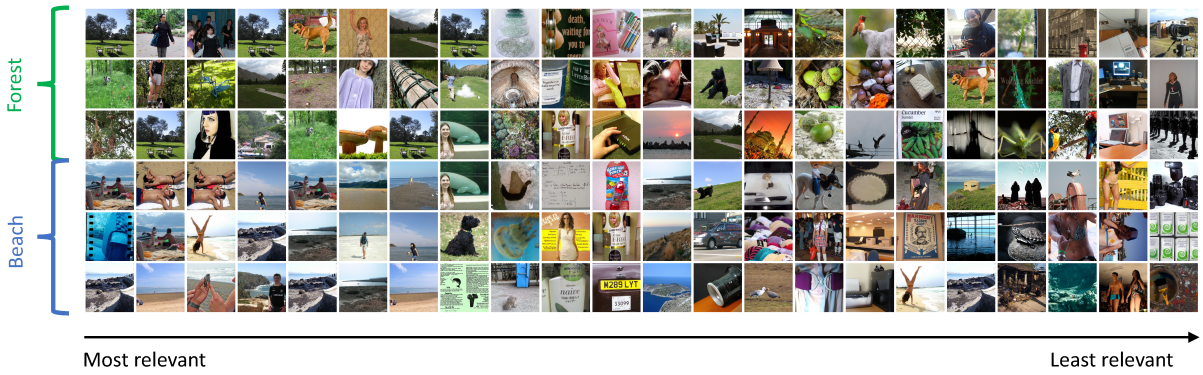

Figure 3: Top-3 images retrieved by DeiT components for "forest" and "beach" ordered according to their relevance for the attribute "location". Each column here corresponds to the images returned by the sum of contributions of 3 components, so column i corresponds to components $\boldsymbol{c}_{3i}, \boldsymbol{c}_{3i+1}, \boldsymbol{c}_{3i+2}$. A large fraction of components which can recognize the "location" feature are sorted correctly by the scoring function

and DINOv2; 134 for SWIN, and 156 for MaxVit.

We then ask if it is possible to attribute a feature-specific role to each component using an algorithm such as TEXTSPAN (Gandelsman et al., 2024). These image features may be low-level (shape, color, pattern) or high-level (such as location, person, animal). However, such roles are not necessarily localized to a single component, but may be distributed among multiple components. Furthermore, each individual component by itself may not respond significantly to a particular feature, but it may jointly contribute to identifying a feature along with other components. Thus, rather than rigidly matching each component with a role, we aim to devise a *scoring function* which can assign a score to each component - feature pair, which signifies of how important the component is for identifying a given image feature. A continuous scoring function allows us to select multiple components relevant to the feature by sorting the components according to their score.

We devise this scoring function (described in the appendix in Alg. 2) by looking at the projection of each contribution vector $\boldsymbol{c}_i$ onto a vector space corresponding to a certain feature. Suppose we have a feature, "pattern", that we want to attribute to the components. We first describe the feature in terms of an example set of feature *instantiations*, such as "spotted", "striped", "checkered", and so on. We then embed each of these texts to CLIP space, obtaining a set of embeddings B. We also calculate the CLIP aligned contributions $f_i(\boldsymbol{c}_i)$ for each component i over an image dataset (ImageNet-1k validation split). Then, the score is simply the correlation between projection of $f_i(\boldsymbol{c}_i)$ and the projection of $\sum_i f_i(\boldsymbol{c}_i)$ onto the vector space spanned by B. Intuitively, this measures how closely the component's contribution correlates with the overall representation. The scores obtained for each component and feature can be used to rank the components according to its importance for a given feature to obtain a *component ordering*, or to rank the features according to its importance for a specific component to get a *feature ordering* .

Text based image retrieval: We can now use our framework to identify components which can retrieve images possessing a certain feature most effectively. Using the scoring function described above, we can identify the top k components $\{\boldsymbol{c}_i\}_{i=1}^k$ which are the most responsive to a given feature p. We can use the cosine similarity of $\sum_{i=1}^k f_i(\boldsymbol{c}_i)$ to the CLIP embedding of an instantiation s_p of the feature p to retrieve the closest matches in ImageNet-1k validation split. In Fig. 3, we show the top 3 images retrieved by different components of the DeiT model for the location instantiation "forest" and "beach" when sorted according to the component ordering for the "location" feature. As the component score decreases, the images retrieved by the components grow less relevant. Also note that a significant fraction of components are capable of retrieving relevant images. This further confirms the need for a continuous scoring function which can identify multiple components relevant to a feature.

To quantitatively verify our scoring function, we devise the following experiment. We first choose a set of common image features such as color, pattern, shape, and location, with a repre-

Model	Feature ordering	Component ordering
DeiT	0.531	0.684
DINO	0.714	0.723
DINOv2	0.716	0.703
SWIN	0.628	0.801
MaxVit	0.681	0.849

Table 2: Spearman's rank correlation between the orderings induced by CLIP score and component score averaged over a selection of common features

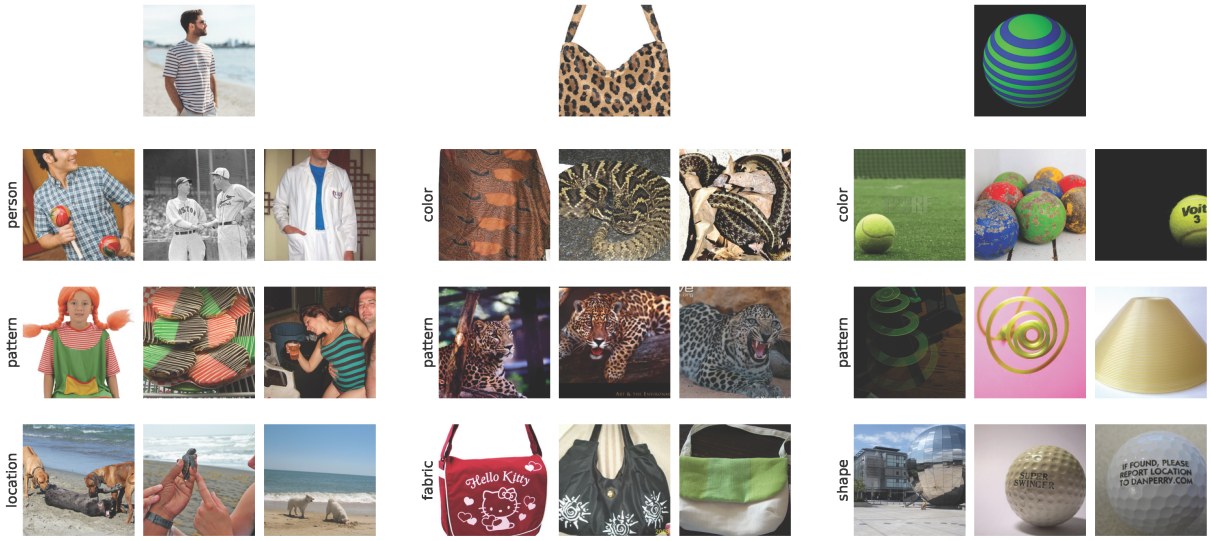

Figure 4: Top-3 images retrieved by the most significant components for various features relevant to the reference image (displayed on top). The models used are (from left to right) DINO, DeiT, and SWIN.

sentative set of feature instantiations for each (details in appendix B). The scoring function induces a component ordering for each feature p and feature ordering for each component i. We then compute the cosine similarity $\text{sim}_{i,s_p} = \cos(f_i(\boldsymbol{c}_i), \boldsymbol{y}_{s_p,\text{CLIP}})$ where $\boldsymbol{y}_{s_p,\text{CLIP}}$ is the CLIP text embedding of s_p. We can compare this to the cosine similarity $\text{sim}_{\text{CLIP},s_p} = \cos(\boldsymbol{z}_{\text{CLIP}}, \boldsymbol{y}_{s_p,\text{CLIP}})$ where $\boldsymbol{z}_{\text{CLIP}}$ is the CLIP image representation. The correlation coefficient between sim_{i,s_p} and $\text{sim}_{\text{CLIP},s_p}$ over an image dataset can be viewed as another score which is purely a function of how well the component i can retrieve images matching s_p as judged by CLIP. Averaging this correlation coefficient over all s_p for a given p yields a "ground truth" proxy for our scoring function. We can measure the Spearman rank correlation (which ranges from -1 to 1) between the component (or feature) ordering induced by our scoring function and the ground truth and average it over features (or components). In Tab. 2 and Tab. 4, we observe that the average rank correlation is significantly high for all models for both feature and component ordering.

Image based image retrieval: We can also retrieve images that are similar to a reference image with respect to a specific feature. To do this, we first choose components which are highly significant for the given feature while being comparatively less relevant for other features. Mathematically, for a feature $p \in P$, the set of all relevant features, we want to choose component i with score $s_{i,p}$ such that the quantity $\min_{p' \in P \setminus p} s_{i,p} - s_{i,p'}$ is maximised. Intuitively, we want components which have the highest gap between $s_{i,p}$ and $s_{i,p'}$ where p' can be any other feature. We can then select a set of k such components C_k by sorting over the score gap, and sum them to obtain a feature-specific image representation $\boldsymbol{z}_p = \sum_{i \in C_k} \boldsymbol{c}_i$. Now, we can retrieve any image $\boldsymbol{x}'$ similar in feature p to a reference image $\boldsymbol{x}$ by computing the cosine similarity between $\boldsymbol{z}'_p$ and $\boldsymbol{z}_p$, which are the feature-specific image representations for $\boldsymbol{x}'$ and $\boldsymbol{x}$. We show a few examples for image based image retrieval in Fig. 4. Here, we tune k to ensure that it is not so small that the retrieved images do not resemble the reference image at all, and not so large that the retrieved images are overall very similar to the reference image. We can see that the retrieved images are significantly similar to the reference image with respect to the given feature, but not similar overall. For example, when the reference image is a handbag with a leopard print, the "pattern" components retrieve images of leopards which have the same pattern, while the "fabric" components return other bags which are made of similar glossy fabric. Similarly, for the ball with a spiral pattern on it, we retrieve images which resemble the spiral pattern in the second row, while they resemble the shape in the third row.

Note that this experiment only involves the alignment procedure for computing the scores and thereby selecting the component set C_k. The process of retrieving the images is based on $\boldsymbol{z}_p$ which exists in the model representation space and not CLIP space. This shows that the model inherently has components which (while not constrained to a single role) are specialized for certain properties, and this specialization is not a result of the CLIP alignment procedure.

Visualizing token contributions: The contribution from a component i can be further decomposed as a sum over con-

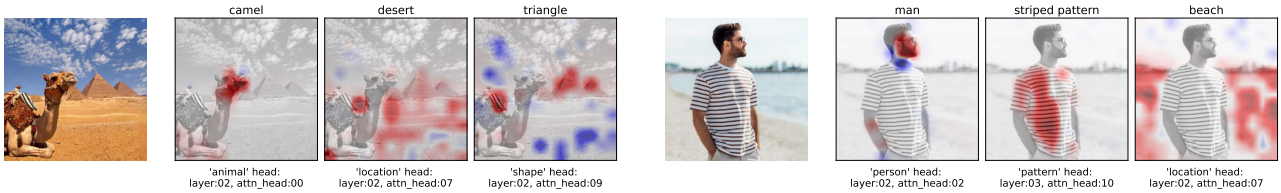

Figure 5: Visualization of token contributions as heatmaps for two example images for the DeiT model. The relevant feature and the head most closely associated with the feature is displayed on the bottom of the heatmap, while the feature instantiation is displayed on the top. The layer numbering starts from the last layer (which has index '00'). The regions highlighted in red contribute positively to the prediction, while blue regions contribute negatively.

tributions from a tokens, so $\boldsymbol{c}_i = \sum_t \boldsymbol{c}_{i,t}$. For any particular CLIP text embedding vector $\boldsymbol{u}$ corresponding to a realization of some feature p, we have $\boldsymbol{u}^\top f_i(\boldsymbol{c}_i) = \sum_t \boldsymbol{u}^\top f_i(\boldsymbol{c}_{i,t})$. We can visualize this token-wise score $\boldsymbol{u}^\top f_i(\boldsymbol{c}_{i,t})$ as a heat map to know which tokens are the most influential with respect to $\boldsymbol{u}$. We show the heat map obtained via this procedure in Fig. 5 for two example images for the DeiT model. The components used for each heat map correspond to the feature being highlighted and are selected using the scoring function we described previously. We can observe that the heatmaps are localized within image portions which correspond to the text description.

Spurious correlation mitigation: We can also use the scoring function to mitigate spurious correlations in the Waterbirds dataset (Sagawa et al., 2020) in a zero-shot manner. Waterbirds dataset is a synthesized dataset where images of birds commonly found in water ("waterbirds") and land ("landbirds") are cut out and pasted on top of images of land and water background. For this experiment, we regenerate the Waterbirds dataset following Sagawa et al. (2020) but take care to discard background images with birds and thus eliminate label noise. We select the top 10 components for each model which are associated with the "location" feature but not with the "bird" class following the method we used for image-based image retrieval. We then ablate these components by setting their value to their mean over the Waterbirds dataset. In Tab. 3, we observe a significant increase in the worst group accuracy for all models, accompanied with an increase in the average group accuracy as well.

Model name	Worst group accuracy	Average group accuracy
DeiT	0.733 → **0.815**	0.874 → **0.913**
CLIP	0.507 → **0.744**	0.727 → **0.790**
DINO	0.800 → **0.911**	0.900 → **0.938**
DINOv2	0.967 → **0.978**	0.983 → **0.986**
SWIN	0.834 → **0.871**	0.927 → **0.944**
MaxVit	0.777 → **0.814**	0.875 → **0.887**

Table 3: Worst group accuracy and average group accuracy for Waterbirds dataset before and after intervention for various models (format is before → **after**)

6. Related Work

Several studies attempt to elucidate model predictions by analyzing either a subset of input example through heatmaps (Selvaraju et al., 2016; Smilkov et al., 2017) or a subset of training examples (Koh & Liang, 2020; Park et al., 2023), however, these approaches are often unreliable in real-world scenarios (Kindermans et al., 2017). These methods do not interpret model predictions in relation to the model's internal mechanisms, which is essential for gaining a deeper understanding of the reliability of model outputs.

Internal Mechanisms of Vision Models: Our work is closely related to the studies by Gandelsman et al. (2024) and Vilas et al. (2023), both of which analyze vanilla ViTs in terms of their components and interpret them using either CLIP text encoders or pretrained ImageNet heads. Like these studies, our research can be situated within the emerging field of representation engineering (Zou et al., 2023) and mechanistic interpretability (Cammarata et al., 2020; Bricken et al., 2023). Other works (Bau et al., 2020; Goh et al., 2021; Olah et al., 2018) focus on interpreting individual neurons to understand vision models' internal mechanisms. However, these methods often fail to break down the model's output into its subcomponents, which is crucial for understanding model reliability. Shah et al. (2024) examine the direct effect of model weights on output, but do not study the fine-grained role of these components in building the final image representation.

Interpreting models using CLIP: Many recent works utilize CLIP to interpret models via text. Moayeri et al. (2023) align model representations to CLIP space with a linear layer, but it is limited to only the final representation and can not be applied to model components. Oikarinen & Weng (2023) annotate individual neurons in CNNs via CLIP, but their method cannot be extended easily to high-dimensional component vectors.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

This project was supported in part by a grant from an NSF CAREER AWARD 1942230, ONR YIP award N00014-22-1-2271, ARO's Early Career Program Award 310902-00001, HR00112090132 (DARPA/ RED), HR001119S0026 (DARPA/ GARD), Army Grant No. W911NF2120076, the NSF award CCF2212458, NSF Award No. 2229885 (NSF Institute for Trustworthy AI in Law and Society, TRAILS), an Amazon Research Award and an award from Capital One.

Author contributions

Sriram Balasubramanian conceived the main ideas, implemented the algorithms, conducted the experiments, and contributed to writing the paper. Samyadeep Basu contributed to the writing and provided essential advice on the presentation and direction of the paper. Soheil Feizi offered critical guidance on the presentation, writing, and overall direction of the paper.

References

Bau, D., Zhu, J., Strobelt, H., Lapedriza, À., Zhou, B., and Torralba, A. Understanding the role of individual units in a deep neural network. *CoRR*, abs/2009.05041, 2020. URL https://arxiv.org/abs/2009.05041.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Cammarata, N., Carter, S., Goh, G., Olah, C., Petrov, M., Schubert, L., Voss, C., Egan, B., and Lim, S. K. Thread: Circuits. *Distill*, 2020. doi: 10.23915/distill.00024. https://distill.pub/2020/circuits.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., and Jitsev, J. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2818–2829, 2023.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

Gandelsman, Y., Efros, A. A., and Steinhardt, J. Interpreting CLIP's image representation via text-based decomposition. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5Ca9sSzuDp.

Goh, G., †, N. C., †, C. V., Carter, S., Petrov, M., Schubert, L., Radford, A., and Olah, C. Multimodal neurons in artificial neural networks. *Distill*, 2021. doi: 10.23915/distill.00030. https://distill.pub/2021/multimodal-neurons.

Ilharco, G., Wortsman, M., Wightman, R., Gordon, C., Carlini, N., Taori, R., Dave, A., Shankar, V., Namkoong, H., Miller, J., Hajishirzi, H., Farhadi, A., and Schmidt, L. Openclip, July 2021. URL https://doi.org/10.5281/zenodo.5143773. If you use this software, please cite it as below.

Kindermans, P.-J., Hooker, S., Adebayo, J., Alber, M., Schütt, K. T., Dähne, S., Erhan, D., and Kim, B. The (un)reliability of saliency methods, 2017.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions, 2020.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Moayeri, M., Rezaei, K., Sanjabi, M., and Feizi, S. Text-to-concept (and back) via cross-model alignment, 2023.

Oikarinen, T. and Weng, T.-W. Clip-dissect: Automatic description of neuron representations in deep vision networks, 2023.

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., and Mordvintsev, A. The building blocks of interpretability. *Distill*, 2018. doi: 10.23915/distill.00010. https://distill.pub/2018/building-blocks.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H. V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D.,

Massa, F., El-Nouby, A., Howes, R., Huang, P.-Y., Xu, H., Sharma, V., Li, S.-W., Galuba, W., Rabbat, M., Assran, M., Ballas, N., Synnaeve, G., Misra, I., Jegou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. Dinov2: Learning robust visual features without supervision, 2023.

Park, S. M., Georgiev, K., Ilyas, A., Leclerc, G., and Madry, A. Trak: Attributing model behavior at scale, 2023.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization, 2020.

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., and Batra, D. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016. URL http://arxiv.org/abs/1610.02391.

Shah, H., Ilyas, A., and Madry, A. Decomposing and editing predictions by modeling model computation, 2024.

Smilkov, D., Thorat, N., Kim, B., Viégas, F. B., and Wattenberg, M. Smoothgrad: removing noise by adding noise. *CoRR*, abs/1706.03825, 2017. URL http://arxiv.org/abs/1706.03825.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention, 2021.

Tu, Z., Talebi, H., Zhang, H., Yang, F., Milanfar, P., Bovik, A., and Li, Y. Maxvit: Multi-axis vision transformer. *ECCV*, 2022.

Vilas, M. G., Schaumlöffel, T., and Roig, G. Analyzing vision transformers for image classification in class embedding space, 2023.

Wightman, R. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., Dombrowski, A.-K., Goel, S., Li, N., Byun, M. J., Wang, Z., Mallen, A., Basart, S., Koyejo, S., Song, D., Fredrikson, M., Kolter, J. Z., and Hendrycks, D. Representation engineering: A top-down approach to ai transparency, 2023.

A. Limitations

Our analysis is limited in several ways which we hope to address in future work. Firstly, similar to Gandelsman et al. (2024), we only consider the direct contributions from the last few layers, and do not look at the indirect contributions though other components. Secondly, we limit ourselves to decomposition only over attention heads and tokens, while convolutional blocks are not decomposed even if they might admit one. Furthermore, it is still unclear if we can identify certain directions or vector subspaces in the model component contributions which are strongly associated with a certain property. We believe that a detailed analysis of higher order contributions with a more fine-grained decomposition may be key for addressing these challenges.

B. Implementation details

Feature instantiations: We use the following features and corresponding feature instantiations. They are chosen arbitrarily:

1. **color**: "blue color", "green color", "red color", "yellow color", "black color", "white color"
2. **texture**: "rough texture", "smooth texture", "furry texture", "sleek texture", "slimy texture", "spiky texture", "glossy texture"
3. **animal**: "camel", "elephant", "giraffe", "cat", "dog", "zebra", "cheetah"
4. **person**: "face", "head", "man", "woman", "human", "arms", "legs"
5. **location**: "sea", "beach", "forest", "desert", "city", "sky", "marsh"
6. **pattern**: "spotted pattern", "striped pattern", "polka dot pattern", "plain pattern", "checkered pattern"
7. **shape**: "triangular shape", "rectangular shape", "circular shape", "octagon"
8. **fabric**: "linen", "velvet", "cotton", "silk", "chiffon"

Hyperparameters: The aligners are trained with learning rate $= 3 \times 10^{-4}$, $\lambda = 1/768$ using the Adam optimizer (with default values for everything else) for upto an epoch on ImageNet validation split. Hyperparameters were loosely tuned for the DeiT model using the cosine similarity as a metric, and then fixed for the rest of the models. We may achieve better performance with more rigorous tuning. The number of components k used for the image-based image retrieval experiment was tuned on an image-by-image basis. It is approximately around 9 for larger models like Swin or MaxVit, and around 3 for the rest.

Computational resources: The bulk of computation is utilized to compute component contributions and train the aligner. Most of the experiments in the paper were conducted on a single RTXA5000 GPU, with 32GB CPU memory and 4 compute nodes.

C. Proof of Theorem 1

Proof. From the first condition on **intra-component rank ordering**, for any two vectors $\boldsymbol{u}, \boldsymbol{v}$ and a linear map f_i, if $\|\boldsymbol{u}\| \leq \|\boldsymbol{v}\|$ then $\|f_i(\boldsymbol{u})\| \leq \|f_i(\boldsymbol{v})\|$. We first show that f_i is a scalar multiple of an isometry.

If $\|\boldsymbol{u}\| = \|\boldsymbol{v}\| \neq 0$, then both $\|\boldsymbol{u}\| \leq \|\boldsymbol{v}\|$ and $\|\boldsymbol{v}\| \leq \|\boldsymbol{u}\|$. This implies that $\|f_i(\boldsymbol{u})\| \leq \|f_i(\boldsymbol{v})\|$ and $\|f_i(\boldsymbol{v})\| \leq \|f_i(\boldsymbol{u})\|$. Therefore, $\|f_i(\boldsymbol{u})\| = \|f_i\boldsymbol{v}\|$, when $\|\boldsymbol{u}\| = \|\boldsymbol{v}\|$. Given the input space of the transformation as U, we choose a unit vector $\boldsymbol{u}_{\text{unit}} \in U$. Let's assume $\|f_i(\boldsymbol{u}_{\text{unit}})\| = c$. With the above result, we can use the following equality $\|\frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}\| = \|\boldsymbol{u}_{\text{unit}}\|$ to obtain the following:

$$\left\| \frac{f_i(\boldsymbol{u})}{\|\boldsymbol{u}\|} \right\| = \left\| f_i\left(\frac{\boldsymbol{u}}{\|\boldsymbol{u}\|} \right) \right\| = \|f_i(\boldsymbol{u}_{\text{unit}})\| = c, \tag{1}$$

Therefore:

$$\|f_i(\boldsymbol{u})\| = c\|\boldsymbol{u}\| \tag{2}$$

Thus, the linear transformation f_i is a scalar multiple of an isometry. Now consider two linear maps f_i and f_j such that $\|f_i(\boldsymbol{u})\| = c_i\|\boldsymbol{u}\|$ and $\|f_j\boldsymbol{u}\| = c_j\|\boldsymbol{u}\|$. From the second condition on **inter-component rank ordering**, for any two vectors $\boldsymbol{u}, \boldsymbol{v}$ and linear maps f_i, f_j, if $\|\boldsymbol{u}\| \leq \|\boldsymbol{v}\|$ then $\|f_i(\boldsymbol{u})\| \leq \|f_j(\boldsymbol{v})\|$. This implies that if $\boldsymbol{u} = \boldsymbol{v}$, $\|f_i(\boldsymbol{u})\| = \|f_j(\boldsymbol{u})\|$. However, this can only happen when $\|f_i(\boldsymbol{u})\| = c\|\boldsymbol{u}\|$ for some constant c for all f_i $\forall i$.

With this, let's denote $\frac{f_i}{c}$ as an isometry. One of the general property of isometries are that they preserve the inner product between two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. First we prove that isometries preserve the inner product, which we will then use to prove the orthogonality of $\frac{f_i}{c}$. Given two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, their inner product can be expressed as the following:

$$\boldsymbol{u}^T\boldsymbol{v} = \frac{1}{4}(\|\boldsymbol{u}+\boldsymbol{v}\|^2 + \|\boldsymbol{u}-\boldsymbol{v}\|^2) \tag{3}$$

An isometry by definition preserves the norm of the vectors i.e. $\|f_i(\boldsymbol{u})\| = \|\boldsymbol{u}\|$ and $\|f_i(\boldsymbol{v})\| = \|\boldsymbol{v}\|$. Due to this property, we can express the following relations:

$$\|f_i(\boldsymbol{u}+\boldsymbol{v})\| = \|\boldsymbol{u}+\boldsymbol{v}\|, \tag{4}$$

and

$$\|f_i(\boldsymbol{u}-\boldsymbol{v})\| = \|\boldsymbol{u}-\boldsymbol{v}\|, \tag{5}$$

We can express $f_i(\boldsymbol{u})^T f_i(\boldsymbol{v})$ as the reduction from Eq.(3):

$$f_i(\boldsymbol{u})^T f_i(\boldsymbol{v}) = \frac{1}{4}(\|f_i(\boldsymbol{u})+f_i(\boldsymbol{v})\|^2 + \|f_i(\boldsymbol{u})-f_i(\boldsymbol{v})\|^2), \tag{6}$$

$$f_i(\boldsymbol{u})^T f_i(\boldsymbol{v}) = \frac{1}{4}(\|f_i(\boldsymbol{u}+\boldsymbol{u})\|^2 + \|f_i(\boldsymbol{u}-\boldsymbol{v})\|^2), \tag{7}$$

Next we substitute the relations from Eq.(4) and Eq.(5) to Eq.(7) to obtain the following inner product preservation property:

$$f_i(\boldsymbol{u})^T f_i(\boldsymbol{v}) = \frac{1}{4}(\|\boldsymbol{u}+\boldsymbol{v}\|^2 + \|\boldsymbol{u}-\boldsymbol{v}\|^2) = \boldsymbol{u}^T\boldsymbol{v} \tag{8}$$

Next we use the inner product preservation property to prove the orthogonality of $\frac{f_i}{c}$ as follows:

$$f_i(\boldsymbol{u})^\top f_i(\boldsymbol{v}) = c^2\boldsymbol{u}^\top\boldsymbol{v}, \tag{9}$$

$$\boldsymbol{u}^\top\left(\frac{1}{c^2}f_i^\top f_i - I\right)\boldsymbol{v} = 0, \tag{10}$$

From 10, we can infer the orthogonality of $\frac{f_i}{c}$ which leads to the following result:

$$f_i^\top f_i = c^2 I = kI, \tag{11}$$

□

D. Scoring function

Algorithm 2 Scoring function for attributing properties to components

Input: Z, the image representation output by the model over n images with dimension d (shape: $n \times d$); C, the contribution of a particular component of interest (shape: $n \times d$); B, the set of k feature vectors that represent a given feature (shape: $k \times d$)

Output: A score that signifies the importance of the component for the given feature

function COMPATTRIBUTE($\boldsymbol{c}$, $\boldsymbol{z}$, B)

$B \leftarrow$ orthogonalize(B)

$s_Z \leftarrow ZB^\top$

$s_C \leftarrow CB^\top$

$r \leftarrow$ correlation_coefficient(s_Z, s_C, dim=0)

return mean(r)

E. Text-based Image retrieval

Model name	Color	Texture	Animal	Person	Location	Pattern	Shape
DeiT	0.679	0.563	0.774	0.596	0.818	0.597	0.764
DINO	0.663	0.657	0.781	0.742	0.833	0.680	0.706
DINOv2	0.751	0.614	0.875	0.714	0.857	0.597	0.510
SWIN	0.795	0.720	0.904	0.780	0.912	0.760	0.739
MaxVit	0.872	0.832	0.911	0.828	0.901	0.803	0.797

Table 4: Spearman rank correlation for various common properties

F. Image-based Image retrieval

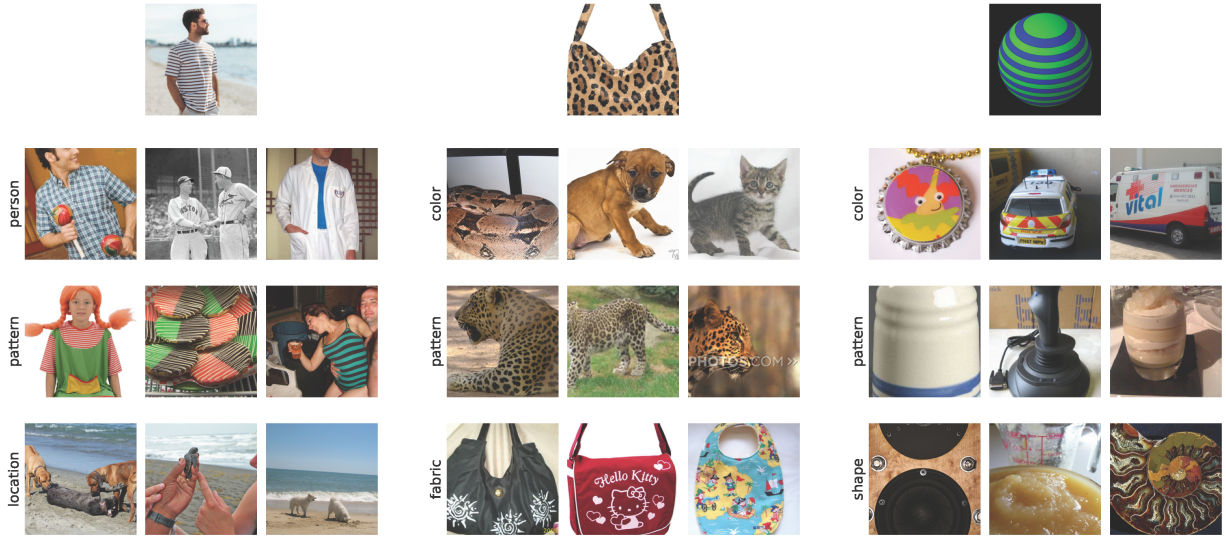

Figure 6: Top-3 images retrieved by the most significant components for various relevant properties for DINO

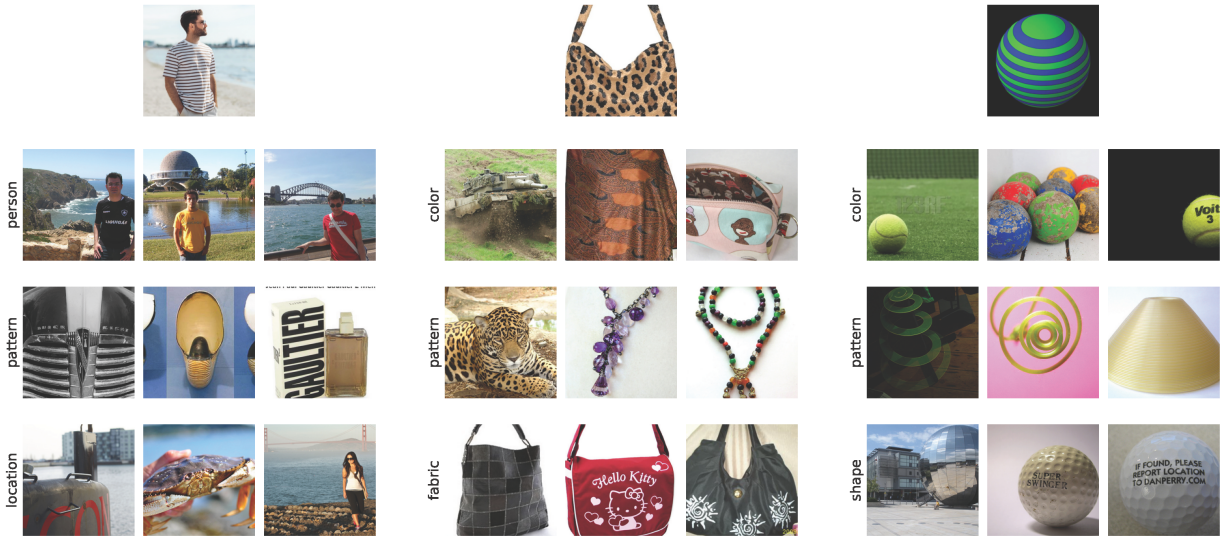

Figure 7: Top-3 images retrieved by the most significant components for various relevant properties for SWIN

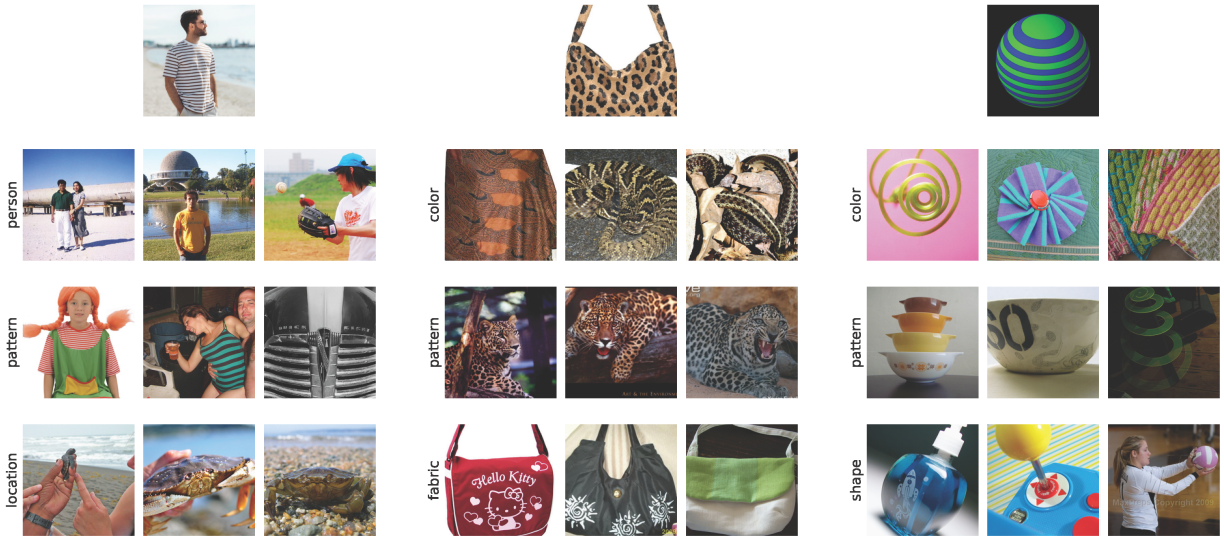

Figure 8: Top-3 images retrieved by the most significant components for various relevant properties for DeiT

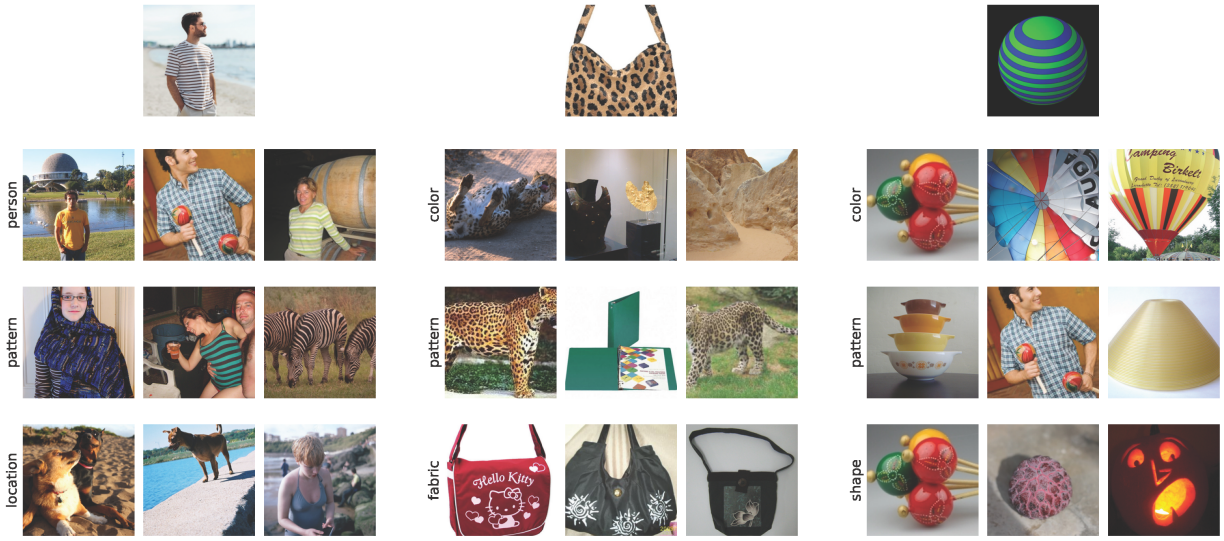

Figure 9: Top-3 images retrieved by the most significant components for various relevant properties for MaxViT

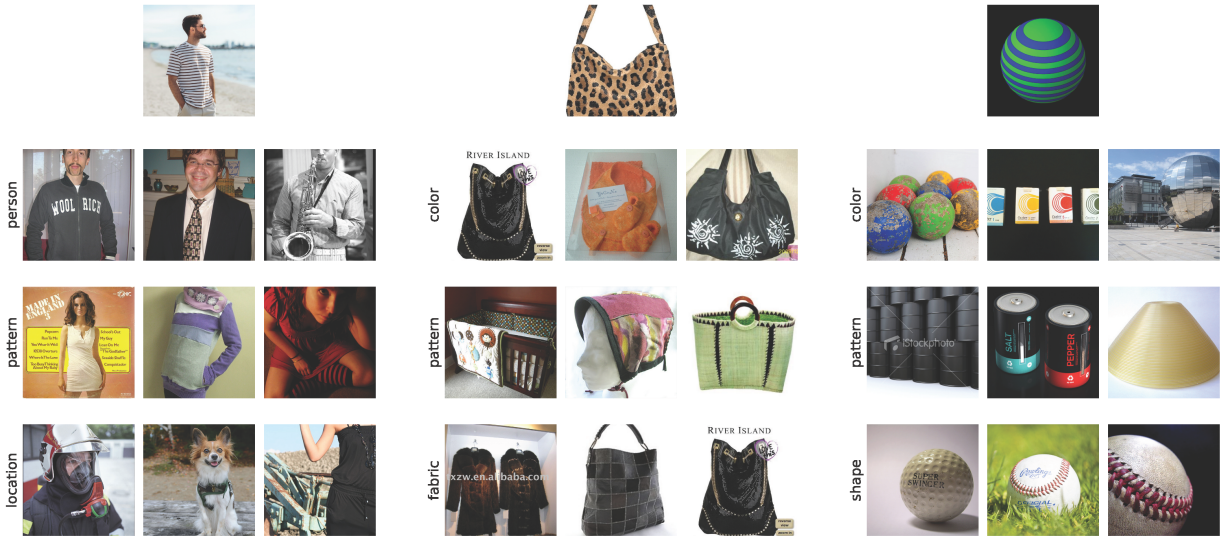

Figure 10: Top-3 images retrieved by the most significant components for various relevant properties for DINOv2

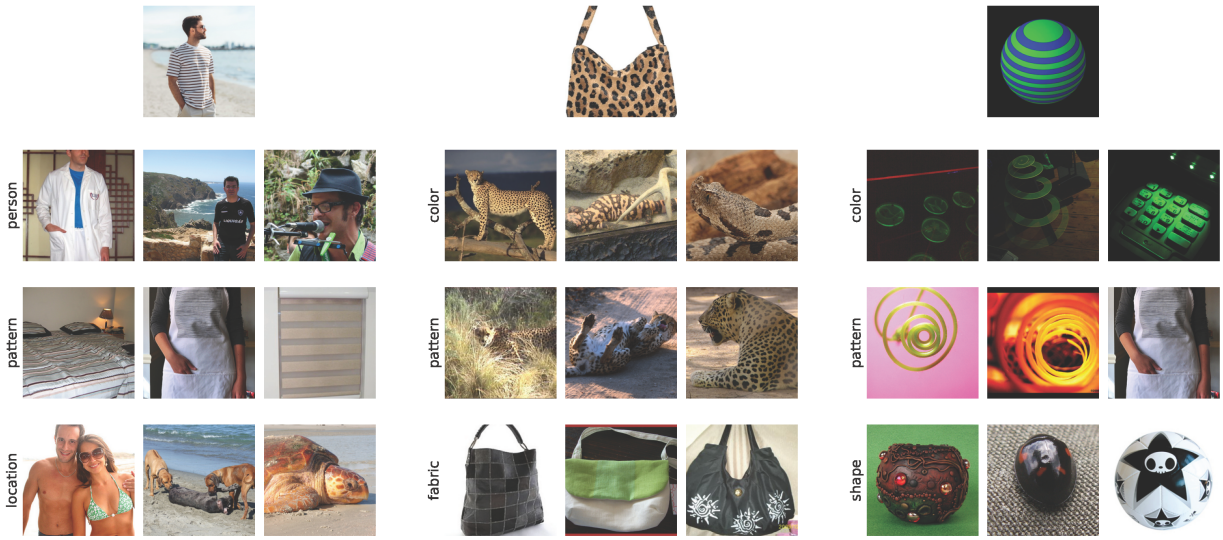

Figure 11: Top-3 images retrieved by the most significant components for various relevant properties for CLIP

G. Property visualization via token decomposition

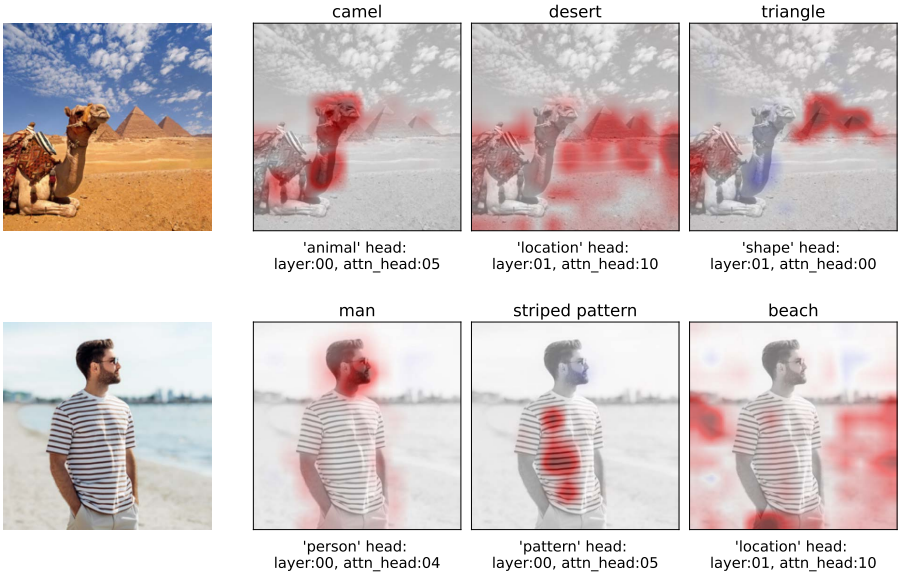

Figure 12: Visualization of token contributions for CLIP

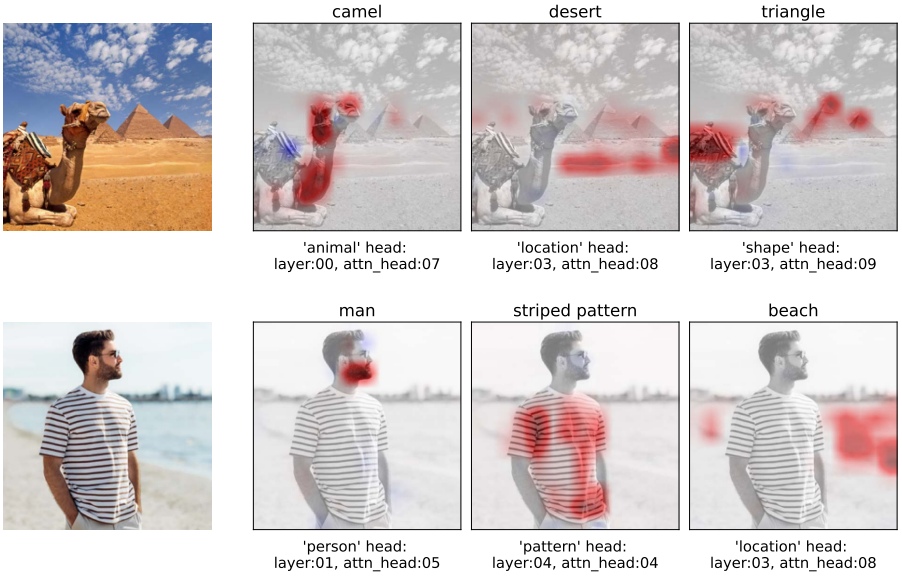

Figure 13: Visualization of token contributions for DINO

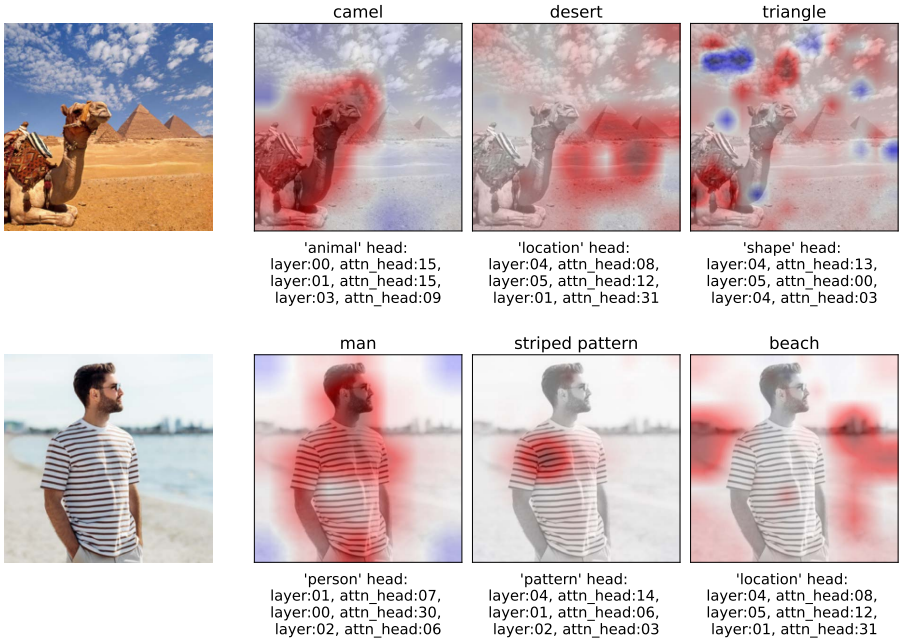

Figure 14: Visualization of token contributions for SWIN

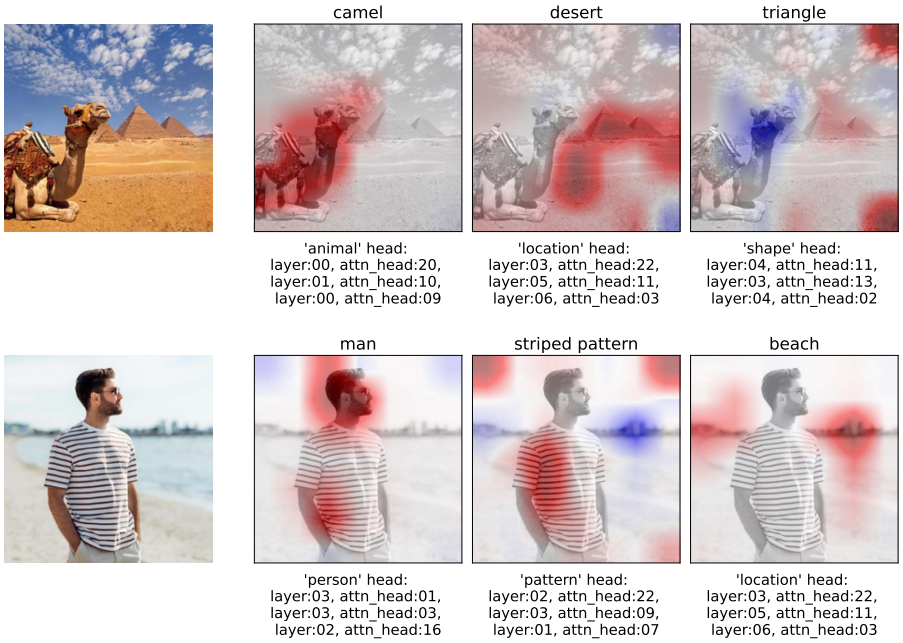

Figure 15: Visualization of token contributions for MaxVit

H. Zero-shot spurious correlation mitigation

Model name	Waterbird in water	Waterbird in land	Landbird in water	Landbird in land
DeiT	0.985 → 0.971	0.733 → 0.815	0.787 → 0.886	0.991 → 0.980
CLIP	0.920 → 0.814	0.507 → 0.746	0.534 → 0.744	0.948 → 0.857
DINO	0.985 → 0.944	0.800 → 0.911	0.832 → 0.943	0.982 → 0.956
DINOv2	0.994 → 0.989	0.967 → 0.981	0.971 → 0.978	1.000 → 0.997
SWIN	0.989 → 0.989	0.834 → 0.871	0.893 → 0.923	0.994 → 0.994
MaxVit	0.959 → 0.942	0.796 → 0.814	0.777 → 0.832	0.970 → 0.961

Table 5: All group accuracies on the Waterbirds dataset before and after component ablation