Topological properties of vision encoders

G., Anton

September 25, 2024

Contents

---

1 Problem statement

Analyse the topological properties of encoders in vision architectures.

Part 1

Evaluate *anisotropy* and *intrinsic dimensionality* of various layers of the visual encodes.

1.1 Consider the following architectures (the list may be extended):

- "ViT-base (openai/clip)" i.e. CLIP (Contrastive Language-Image Pre-Training), openai/clip-vit-base-patch16;
- "convNext-base" i.e. ConvNeXt, convnext-base-224;
- "BEIT-base" i.e. BEiT: BERT Pre-Training of Image Transformers, microsoft/beit-base-patch16-224.

1.2 Select dataset and explain the choice.

1.3 Provide a method for calculating the *intrinsic dimensionality* and *anisotropy*. Explain what is being computed and what is the meaning behind each of the metrics.

1.4 Analyze the results for the architectures under consideration.

Part 2

Evaluate the *internal representations* for the vision models under consideration.

2.1 Perform a *linear probing* procedure for the embeddings from each layer of the vision model.

2.2 The CIFAR-100 dataset is proposed.

2.3 Explain how the input embedding is computed for each model.

2.4 Train the classifier.

2.5 Interpret the results.

2 Topological properties of vision encoder layers (part 1)

2.1 Intrinsic dimension

Intrinsic dimension (ID) refers to the minimum number of variables needed to describe the underlying structure of a dataset i.e. the dimensionality of the data manifold. The ID estimator used in the code is known as the Two Nearest Neighbors (TwoNN) algorithm [1], which leverages the distances between the nearest neighbours of points in a high-dimensional space – embedding dimensionality (ED).

First, we compute the pairwise distances $dist(i,j) = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2}$ where $x_{i,k}$ is the k-th dimension of the point x_i, and d is the dimensionality of the space (ED). For each point i, the distances to the first and second nearest neighbors to be found as r_1 and r_2, respectively.

The implementation excludes the points where $r_1 = 0$ (zero distance to the nearest neighbour) or $r_1 = r_2$ (equidistant neighbors) to avoid problematic points during the computation. After filtering, the ratio of the distcanes is calculated for every point:

$$\mu = \frac{r_2}{r_1}.$$

Next, the empirical distribution F_{emp} is defined as:

$$F_{\text{emp}}(i) = \frac{i}{N}, \quad i = 1, 2, \ldots, N$$

where N is the number of filtered valid points. Then the following transforms are applied:

$$x = \log(\mu), \quad y = -\log(1 - F_{\text{emp}}).$$

Linear regression is then performed between x and y, with the slope being the ID estimate. Finally, the correlation coefficient r and the p-value p are calculated to evaluate the quality of the linear fit and provide insight into the strength of the linear relationship between x and y.

The TwoNN is primarily considered a global method since it attempts to estimate the overall ID of the dataset by utilizing the distances between each data point and its two nearest neighbours across the entire dataset. The TwoNN does not adapt to the local variations of the dataset and assumes that the ID is homogeneous across the entire dataset.

2.2 Anisotropy

Anisotropy refers to the degree to which the variance in the embedding space is concentrated along certain directions and provides insight to whether the embeddings are spread uniformly across all directions (i.e. isotropic) or concentrated along specific directions (i.e. anisotropic).

For a set of embedding vectors represented by a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n is the number of samples and d is the number of features, the anisotropy is defined as the ratio of the largest eigenvalue $\lambda_{\max}$ of the covariance matrix of the embeddings to the sum of all eigenvalues:

$$\text{A} = \frac{\lambda_{\max}}{\sum_{i=1}^{d} \lambda_i}$$

where λ_i are the eigenvalues of the covariance matrix, i.e. the variance in each direction of the embedding space.

To estimate the anisotropy, the following steps were performed in the code implementation.

The mean of the embeddings is subtracted from each embedding vector to obtain a zero-centered dataset in order to ensure that the covariance matrix captures the variance relative to the mean.

The centered embeddings are decomposed using singular value decomposition (SVD) which returns singular values that are the square roots of the eigenvalues of the covariance matrix. $\mathbf{X}_{\text{centered}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ where $\mathbf{\Sigma}$ contains the singular values $\sigma_1, \sigma_2, \ldots, \sigma_d$.

The eigenvalues of the covariance matrix are related to the singular values by:

$$\lambda_i = \frac{\sigma_i^2}{n-1}$$

where n is the number of samples. The eigenvalues describe how much variance is present in each principal direction. Finally, the ratio of the largest eigenvalue λ_{max} to the sum of all eigenvalues showsn the extent to which the variance in the embedding space is concentrated along the principal direction.

The anisotropy can be understood as following. If the value is close to 1, it indicates that most of the variance is captured by a single direction in the embedding space. In this case the data is essentially lying on a lower-dimensional manifold. Vice versa, if the anisotropy is low, it implies that the variance is more spread across multiple directions, saying that embeddings are more isotropic and cover a wider range of dimensions in the embedding space.

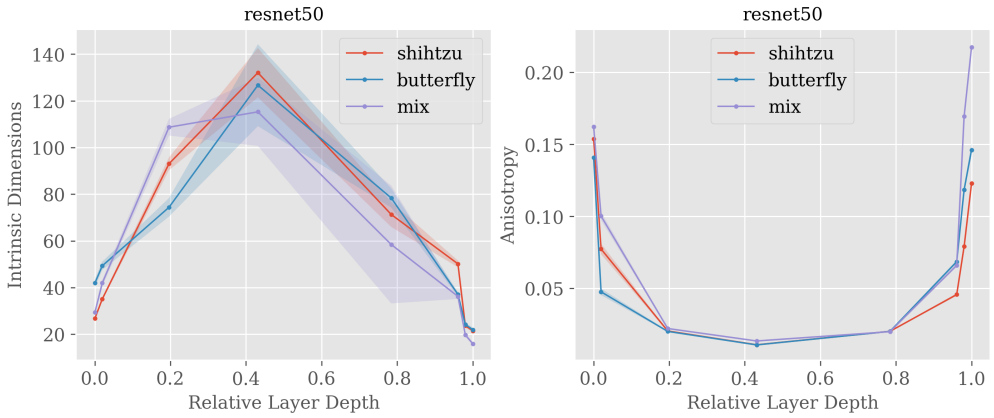
Anisotropic embeddings can indicate that the model focuses on a limited set of features, potentially limiting its ability to generalize. Isotropic embeddings, in contrast, suggest that the model is capturing a more balanced and diverse set of features, leading to a richer data representation.

2.3 Experiment

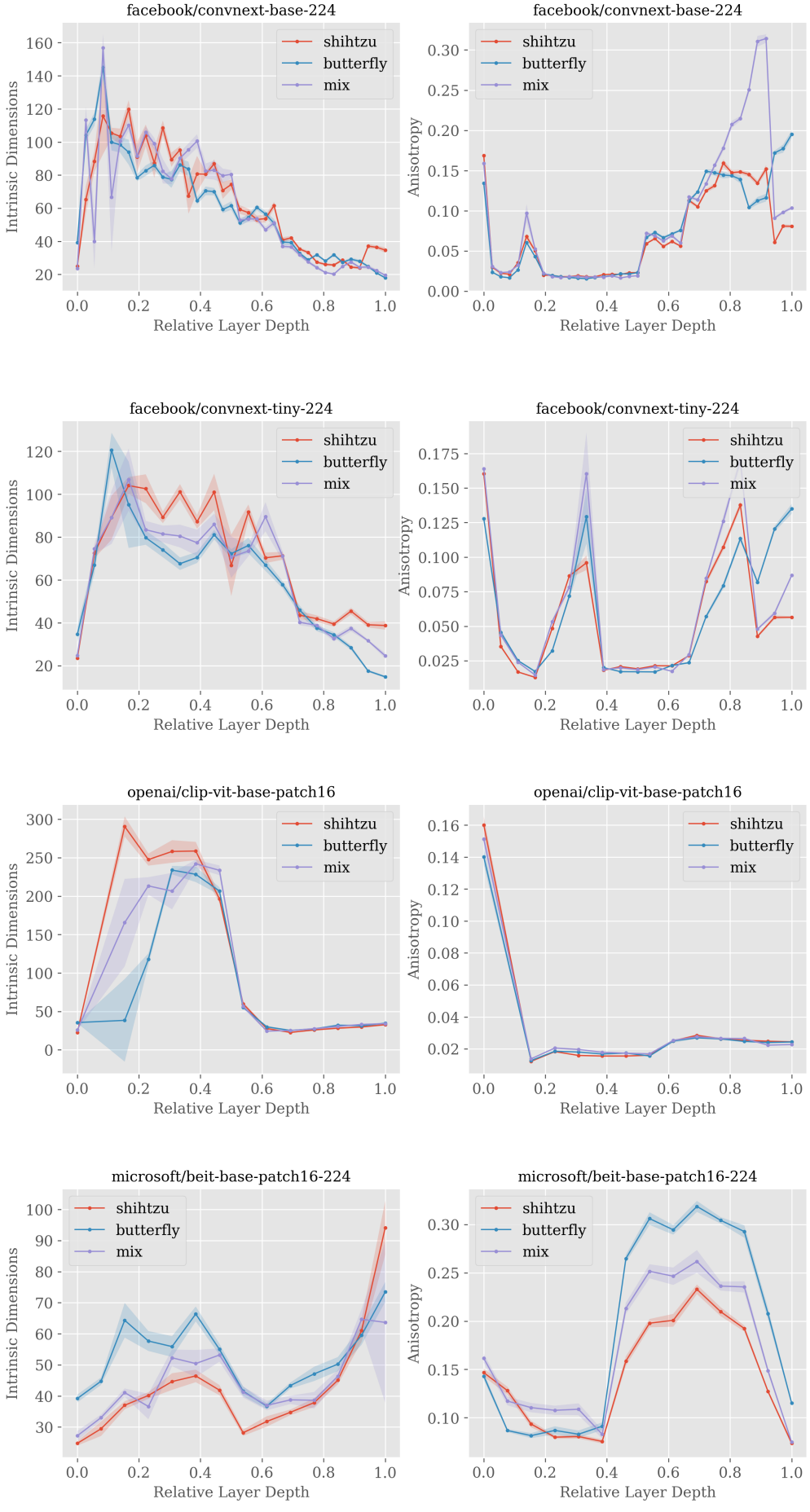
The experimental setup in this study essentially extends to vision transformers the framework developed in [2], where the intrinsic dimensions (IDs) of various CNN architectures were studied. Following the methodology proposed in the original paper, we used randomly sampled 500 images from each of the most popular ImageNet categories to estimate the ID and anisotropy of the resulting object manifolds across the layers of the networks, with each category analyzed independently. In our experiments, we limited the dataset to “shihtzu”, “butterfly”, and “mix” categories chosen to capture different data manifolds across the categories represented on the photos.

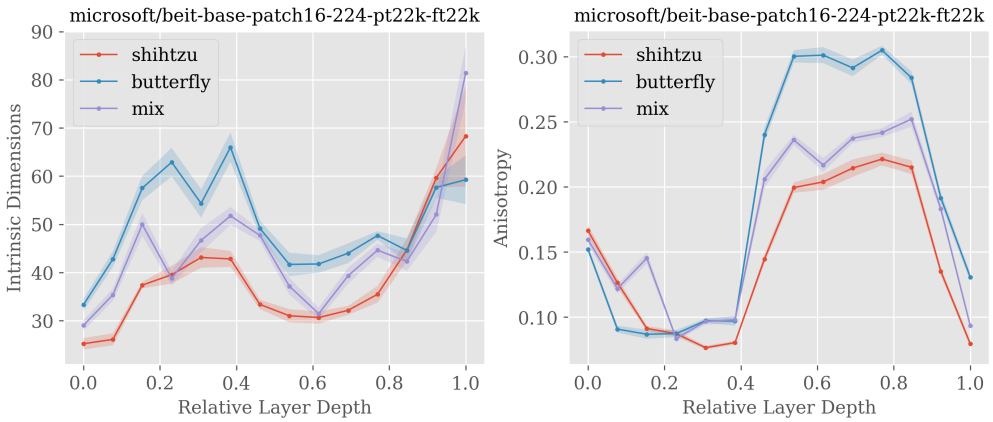
2.4 Results

The experimental outcomes are shown below. First, as a benchmark are taken CNN models:



Then the plots for the vision transformers under consideration:





3 Internal representations of vision encoder layers (part 2)

3.1 Input embeddings

For the Vision Transformer (ViT) i.e. the CLIP model `openai/clip-vit-base-patch16`, the input embedding is computed by first resizing input to a fixed size of 224×224 pixels and normalizing resulting in a tensor of shape $(BS, 3, 224, 224)$. The images are then split into 16×16 patches, resulting in 196 patches for each image. These flattened patches are passed through a linear projection layer to create embeddings of shape $(N, 196, 768)$. Positional encodings of shape $(1, 196, 768)$ are added to keep spatial information.

For ConvNeXt i.e. the `convnext-base-224`, the input embeddings are similarly obtained by resizing and normalizing the input images to a tensor shape of $(BS, 3, 224, 224)$. However, instead of dividing to patches, ConvNeXt uses several convolutional layers. The output after the initial convolutional layers returns a feature tensor shaped as (BS, C, H', W') where C is the number of channels, and H' and W' are the spatial dimensions after downsampling. The feature map is then pooled or flattened, typically resulting in a shape of $(BS, C \cdot H' \cdot W')$ for subsequent downstream tasks.

In BEiT, `microsoft/beit-base-patch16-224` the input images are resized and normalized to the shape $(BS, 3, 224, 224)$. The images are divided into 16×16 patches, resulting in a shape of $(BS, 196, 16 \times 16 \times 3)$. Each patch then tokenized using a pre-trained tokenizer, converting it into discrete tokens, which are then mapped to embeddings through an embedding layer, producing $(N, 196, 512)$. Positional encodings are added to these token embeddings.

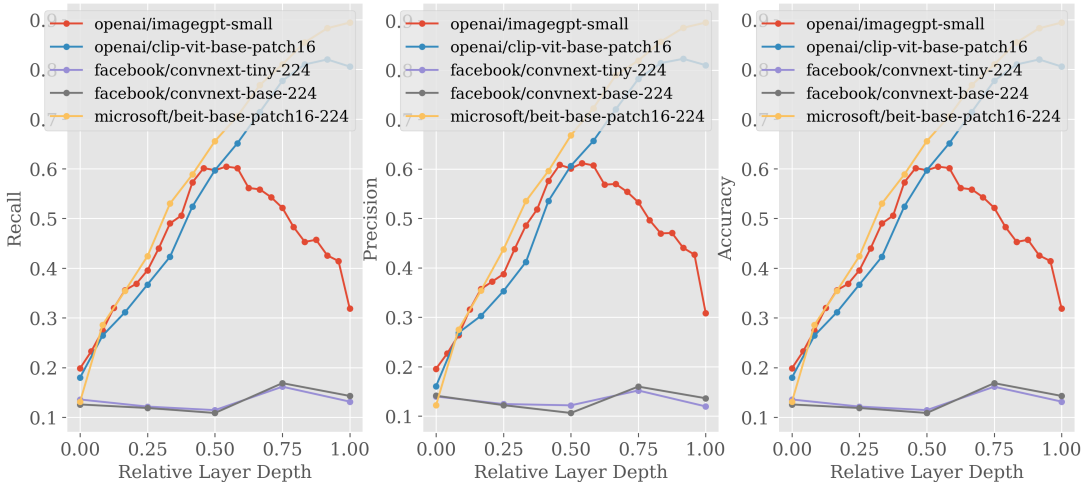
3.2 Linear probing

Linear probing allows to analyze the quality of feature representations learned at different layers of the model. By attaching a linear classifier to the outputs of the layers and evaluating its performance on a specific task (e.g. classification), it is possible to determine how well the representations capture the relevant information needed for that task. Higher accuracy indicates that the features extracted at that layer are more informative for the classification task.

We can also identify which layers contribute the most to task performance. This can help in understanding how information is processed within the model. E.g., first layers may capture low-level features, while deeper layers may learn more abstract representations. Analyzing performance across layers can reveal the depth at which features become most discriminative for a specific tasks.

3.3 Results

As we can see, for the ImageGPT, the best features appear to be obtained in the middle layers, after which performance starts to degrade. In contrast, for ViT-base and BEIT-base, the best features are progressively obtained closer to the output.



The situation is different for ConvNeXt, where the scores remain consistently low without significant improvement. This could be explained by the flattening operation before classification, which collapses the spatial information (i.e., the 2D structure) into a single vector. This process may result in the loss of valuable spatial relationships within the hidden representations, negatively impacting classifier performance.

References

[1] Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio, *Estimating the intrinsic dimension of datasets by a minimal neighborhood information*, Scientific Reports, vol. 7, no. 1, pp. 12140, 2017, Nature Publishing Group UK, London.

[2] Alessio Ansuini, Alessandro Laio, Jakob H. Macke, and Davide Zoccolan, *Intrinsic dimension of data representations in deep neural networks*, Advances in Neural Information Processing Systems, vol. 32, 2019.