Decomposing Image embeddings into Object Embeddings.

Kumar Selvakumaran
MS in Artifical Intelligence
Northeastern University
Boston, USA
selvakumaran.k@northeastern.edu

Mrudula Acharya
MS in Artificial Intelligence
Northeastern University
Boston, USA
acharya.mr@northeastern.edu

Neel Adke
MS in Robotics
Northeastern University
Boston, USA
adke.n@northeastern.edu

***Abstract*—Convolutional Neural Networks are now used unanimously due to their extreme flexibility, scalability, and feature extraction prowess. They gradually encode increasingly complex visual and eventually semantic features as the complexity of the network increases. This behavior of CNNs allows us to extract encoded information at different levels of complexity depending on which parts of the neural network we leverage. Nowadays, Object detection has widespread applications, and many situations may require more nuanced information about objects like color-textural themes (interior design), match-able features (autonomous driving), etc. In this project, We aim to study object-specific embeddings and how they can be used to address the need for subtly defined and contextual features of objects in natural scenes. We also attempt to understand the location-specific information from object embeddings, and how it varies with the change in the location of an object in the image.**

***Index Terms*—Convolutional Neural Networks, Semantic Features, Image Embeddings, Object Embeddings,**

I. Introduction

The demand for powerful semantic feature descriptors for images led to the ubiquity of CNNs, but Training CNNs is a computationally expensive procedure. As a result, there has been extensive research on trying to obtain general-purpose representations of images which can be extended to more tasks than what the CNN was trained on. Such general-purpose representations can be obtained by accessing the treasure trove of information from the input image that is selectively retained and emphasized information by different hidden layers of the neural network. Many studies like [1] have visualized the types of features that are extracted at different hidden layers of a CNN, and a general behavior was noticed where the earlier layers of the neural network capture finer features, and later layers capture coarser features.

This learning behavior of the neural network serves as one of the defining criteria that helps us decide which layers of the neural networks we must choose to get the embedding that is most relevant to our use case. If we choose the layer that is too early in the neural network, we obtain features that are very primitive color-textural features, which are not sufficiently defined semantically, whereas, if we take embeddings that are extremely close to the output layer, in deep networks, We will find that it these embeddings capture almost only class-specific information, and may not be much more helpful than the class output itself.

In this project, We study embeddings from the ResNet50 classification model as discussed in [2], and the YOLOv3 object detection model as discussed in [3]. These studies revolve around studying the information encoded in the embeddings by using them to do K-nearest neighbors matching. Studying an embedding's neighbors, and the ordering of these neighbors provide insights into possible intra-class distinctions in classification models that can be of interest depending on use cases.

We also explore an approach to isolate object-specific embeddings from the image embedding that is extracted from the YOLOv3 model. These object-specific embeddings are again subject to the KNN analysis study. Additionally, We visualize how the embeddings vary with changes in the location of the same object, in an attempt to understand how exactly location information is encoded within an object embedding, and if it can be extracted..

II. Related Work

There have been elaborate studies about how object emeddings can be used to recognize objects in context [4] using embeddings of objects and embeddings of local neighbors of the object crop. crops of furniture in images were used as objects, and a plethora of such instances were analyzed. These embeddings were visualized on a grand scale using the tSNE algorithm. The extent to which the semantic clusters of object embeddings were organized emphasized the descriptive power of these neural network embeddings.

In addition to clear class-specific clustering, there was a presence of nested clusters where furniture with similar design patterns were close to each other in the embedding space. The fact that such nuanced details are encoded by embeddings and how they are semantically organized in the embedding space attests to the richness of information present in embeddings of a neural network. More interestingly, These embeddings are obtained from a neural network that is trained on a task as high-level as classification. Although additional training

procedures are sometimes needed to get a more structured embedding space, these embeddings can be used to accelerate and lighten the computation efforts required for training on further downstream tasks, making these additional efforts worth it.

III. Classification Model Embeddings

We initiated our study with classification embeddings as we don't have to worry about the isolation of any entity. Classification, by nature, focuses on an entity of interest, as a result, to obtain object embeddings, the best way would be to pass an object crop into the neural network so that it is not polluted by any background features that the model may have learned.

A. Embedding extraction

Considering an image from a natural scene with many objects, We run the Grounded SAM model as introduced in [5] to get object crops using zero-shot prompt-based object detection and segmentation. For classification-based embedding experiments, we have used images belonging to 3 different classes namely, airplane, apple, and dog. After the classes were decided, the class prompts were passed accordingly to the Grounded SAM model which then detects all the apples, airplanes, and dogs from the image database as shown in Fig.1. The resulting segmentation masks are used to crop the objects out of the image, and the bounding boxes are used to make smaller auxiliary images with black backgrounds that tightly fit the object. The object crops are then pasted on these auxiliary images as shown in Fig.1, and these resulting auxiliary images are then fed into the classification model for object-specific embedding extraction.

Fig. 1. constructing model input auxilliary image.

These auxiliary images are then fed to the ResNet50 classification model as input, and a 2048 dim flattened embedding is obtained from the penultimate layer of the ResNet model. This embedding is then stored for future computations, along with the corresonding class, bounding box, segmentation, and image details.

B. KNN analysis

The results from the embedding extraction phase is a matrix of shape [nxd] embedding matrix where n is the number of images in the dataset and d is the dimension of the embedding (2048). Now, using Euclidean distance, K nearest neighbors matching is done. Observing the neighbors that are close to a given target embedding, and how these top K neighbors are ordered, describes how the embeddings are organized in the embedding space under the assumption that the space can be described as R^d. The K nearest neighbors are then visualized as a grid with the input auxiliary images that generate the respective embeddings to visually inspect which instances are close to the the target object instance. The most interesting insights derived from KNN analysis are discussed below.

1) orientation information: In the KNN visualization from Fig.2., particularly those of apples, paying attention to the stalk position on the apple suggests the orientation of the apple. From the visualization, it can be observed, that, for all Apple target instances, The model can accurately match the similarity of orientation. Instances with similar stack positions, and directions are the closest matches in the embedding space. Whereas, if there are no similar orientations, the diametrically opposite positions, are the most distant matches.

These results suggest that the model cannot only learn similar orientations, but the orientations in the embedding space seem to represent an abstraction of the polar coordinate system. Although there is not enough evidence to say that this is a property of the entire embedding space, evidence from the visualization suggests that this might be the case for apples at least. Another point to note is that, although the orientation is learned, color features are prioritized more than expected because we don't find any matches of differently colored apples even if they have similar orientations.

2) intra-class semantic differentiation: Although, there isn't much evidence to completely back up this insight. On observing the first target dog instance (1st column, 8th row) we find that the model finds the "Rhodesian ridgeback" class (target) more similar to the "Labrador retriever" class (M 1) than the "kuvasz" class (M 2). This suggests that within the embedded space corresponding to the dog class, there may be a considerable separation between larger/older dogs, and smaller/younger dogs (puppies). This kind of intra-class separation will be especially useful, in in-cases the model needs to be trained on finer-grain classification.

IV. Object Detection Model embeddings

Now that we have an idea of how we can access object-specific semantic information through image embeddings, we aim to extend this technique to the object detection task, which introduces a new task, which is how to isolate individual object embeddings when the model takes a single image and predicts multiple objects (classified bounding boxes).

A. Embedding extraction

To device the procedure used to extract object-specific embeddings, we look deeper into the working of the YOLOv3

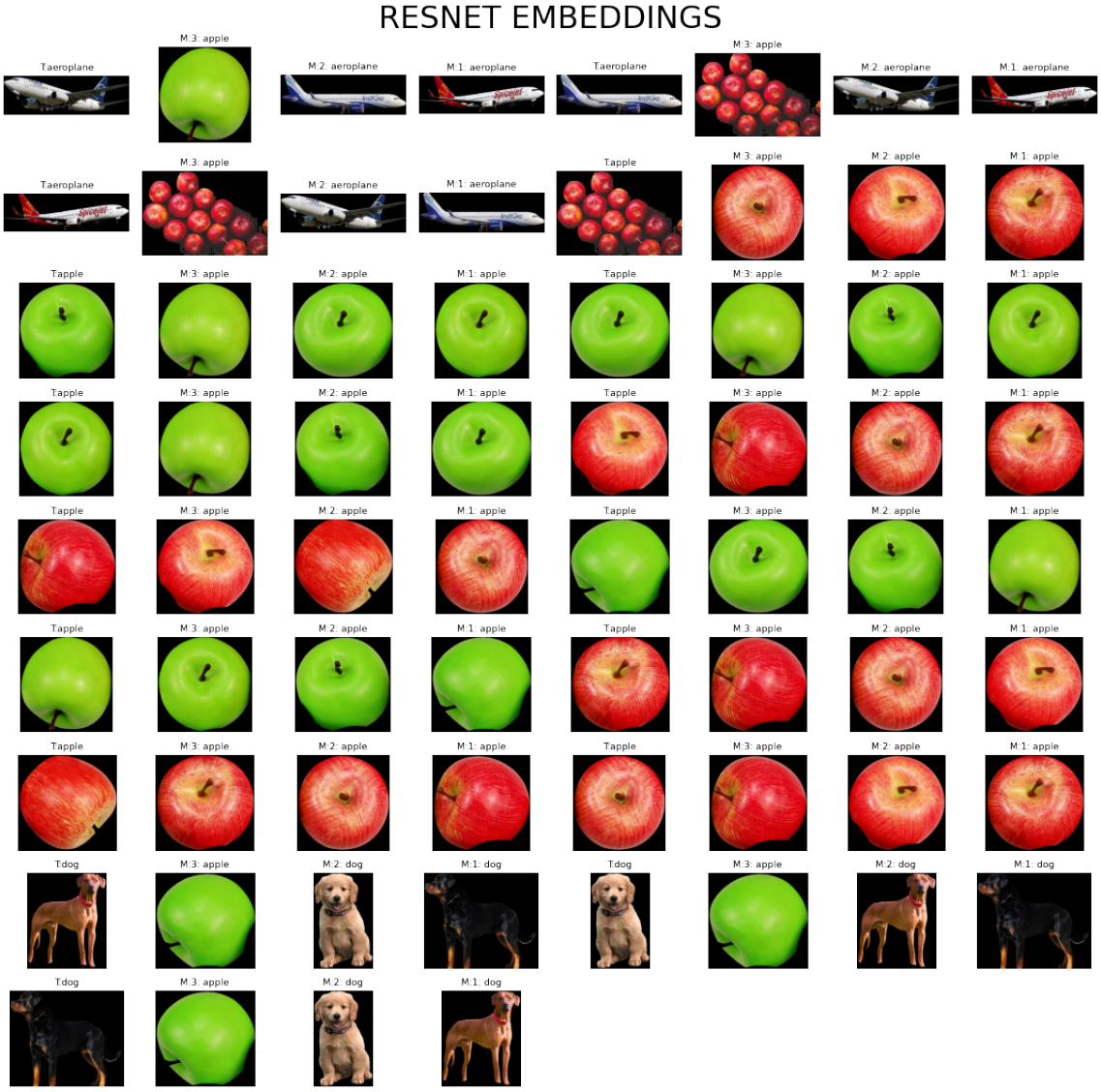

Fig. 2. Nearest neighbours visualization of ResNet object embeddings.

model, more specifically, How the training is supervised for the model.

The procedure implemented was motivated by how ground truth bounding boxes are assigned to the model, while training, is used to incur mis-classifcation and mis-localization loss. The deepest layer of the YOLOv3 model produces a 13x13 embedding with 255 channels. out of these 13x13 grids, the responsibility of predicting the bounding box is assigned to the cell which contains the center of the bounding box. Hence, we can expect that in the trained model, the grid that contains the center of the bounding box will have predicted that the bounding box, and the features corresponding to the object in this bounding box, will have accumulated in the grid that contains the center. In reality, the features accumulate more towards the center of the image, due to padded convolutions, But using the box-center-membership approach proves to be an effective baseline.

We use the 2nd last layer from the YOLOv3 model for extracting the embeddings, and we take the appropriate window that will be convolved into the grid of the final layer that contains the box center. We explore 2 options, extractings of a window size of 5, which should ideally contain all of the information of the object but may also contain some background information, and embeddings from a window size = 2 centered by the grid that contains the center of the bounding box. In this project, We try to isolate information that is as specific to the object as possible, hence we decide to use an embedding window of size 3 as shown in Fig.3, even though we lose some information about the object. The 3x3 window with the brightest grid is the slice taken which exactly corresponds to The patch enclosed by the black box in the image, and hence for the object whose bounding box's center is visualized as a black and white dot.

Experimentation has proved that the information captured by an embedding window of size 3 captures enough information to confidently describe the relations of objects in the embedding space. As seen in Fig.3, We have conducted an embedding occlusion analysis to justify our theoretical grounds. The occlusion analysis is done by zeroing out windows iteratively, over a stride of 1, and increasing the window size each cycle by 1, and seeing how that affects the model's predictions. some small occlusions that were impactful are shown above in Fig.3. The grids marked by X are the grids that were zeroed out. The image on the left corresponds to the complete set of predictions made by the model, and the images on the right correspond to the hindrance of a prediction that was caused by zeroing out a grid.

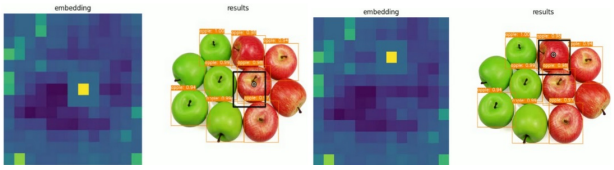

Fig. 3. Isolating object embeddings from the image.

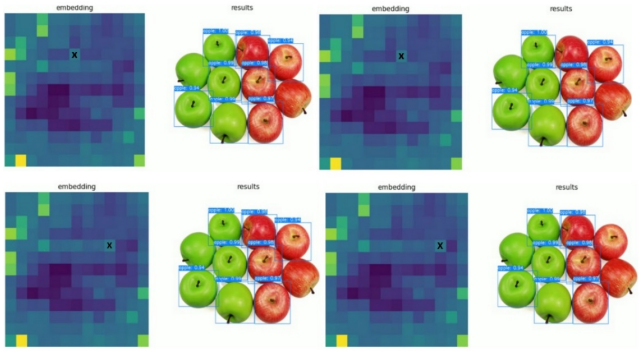

Fig. 4. Constructing model input auxilliary image.

In all images from Fig.3, the heatmap on the left corresponds to the embedding's 1st channel, and the grid marked with 'X' is the grid that was zeroed out in the embedding. In the top right image, we find that occlusion led to the 2nd prediction from the topmost row of predictions disappearing. Similarly, the bottom right image, shows that the occlusion made the rightmost detection disappear. The location of the zero-ed out grid and the corresponding impact in predictions clearly attests to the theoretical ground used to devise the embedding extraction procedure mentioned previously.

Now that we have a clear procedure for isolating embeddings from image embeddings of the YOLOv3 model, we can obtain the corresponding embedding slices, and flatten them, and these will be our new object-specific embedding. For the nearest neighbor analysis of YoloV3's embeddings, a subset of the COCO dataset of the classes 'chair', 'couch', 'potted plant', 'cat', 'dog', and 'person' is used to make the embeddings dataset. The nearest embedding analysis of YOLOv3's embedding space seems to have excellent class-specific separation, but this is unsurprising as it was trained to be a bounding box regressor and classifier. What we are interested in is intra-class semantic distinctions. Although the model doesn't do consistently well on matching similar textures within an instance of the same class, the model seems to exhibit some affinity to contextual information, due to the presence of more than 1 class.

B. KNN analysis

For the nearest neighbor analysis of YoloV3's embeddings, a subset of the COCO dataset of the classes 'chair', 'couch', 'potted plant', 'cat', 'dog', and 'person' is used to make the embeddings dataset. The nearest embedding analysis of YOLOv3's embedding space seems to have excellent class-specific separation, but this is unsurprising as it was trained to be a bounding box regressor and classifier. What we are interested in is intra-class semantic distinctions. Although the model doesn't do consistently well on matching similar textures within an instance of the same class, the model seems to exhibit some affinity to contextual information, due to the presence of more than 1 class.

1) Contextual information/presence of multiple classes: : When multiple classes are present tightly together, this creates some context, that the model picks up on. For instance, consider The target present in the 10th row 11th column of Fig.5, a man sitting on a chair. Both "person" and "chair" are well represented in the COCO dataset, and we can see the model finds images with both classes present similar to the target image. This semantically means "man sitting on a chair" (sitting may be the only situation where the overlap is enough for the model to recognize this context). What is more interesting is that The model doesn't consider situations, where a person is simply standing next to the chair, a few of the person instances in the plot stand close to chairs, but the model has specifically selected situations where the person is sitting on a chair clearly, as neighbors.

2) Semantic similarity across classes: : Consider the object instance in the 6th row, 6th column of Fig.5, It is a leather plush chair. Interestingly, an instance from a different class that looks extremely similar (probably from the same sofa set) to the target is the first embedding neighbor, followed by embeddings of the same class as the target. This suggests that the model is able to object class has not become a hard and fast boundary in the embedding space, and there is still nuanced semantic information and grey areas, that host this kind of similarities.

Shockingly, When it comes to the same class, the model seems to disregard semantic information. One such situation is the target in the 8th row, 8th column of Fig.5, An instance

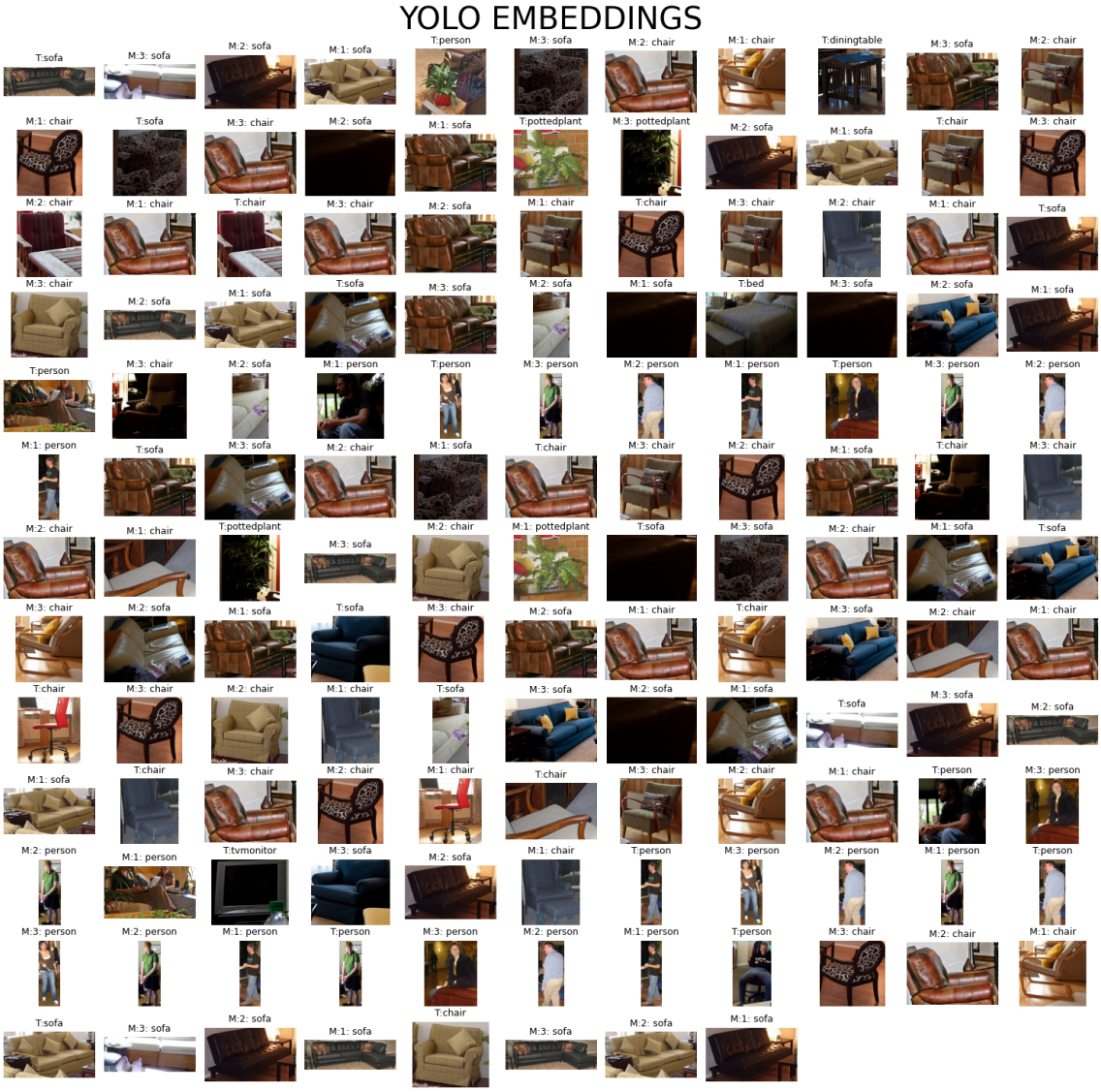

Fig. 5. Nearest neighbours visualization of YOLOv5 object embeddings.

that is a different view of an almost identical object (Match 2) to the target was not matched first. In this case, we would expect match 2 and the target to be extremely close in the embedding space, but this is not the case. This suggests that the embedding space shows some local patterns, but overall is more unstructured than we would like, and difficult to navigate.

C. location analysis

Object detection models act as regressors to predict bounding box coordinates, and classifiers to predict the object classes. This implies that along with semantic information, location, and size information is also encoded in the embeddings. This project also tries to understand how localization and semantic information are jointly encoded, and more specifically, What aspect of the embedding is unique to localization information. To understand and observe location-specific information from object detection embeddings, The approach used is to analyze the the difference in embeddings where the object is identical, but is translated to different parts of the image as shown in Fig.6.

The motivation behind this experiment is that using duplicate translated crops of the same object must entail minimizing the change in semantic information between the corresponding embeddings. The only difference between the 2 corresponding embeddings must arise from the change in position of the object. By studying this change, we hope to better understand the characteristics of localization information's presence in the object embeddings.

The embeddings of the Yolo model are of size 9216, which inhibits our ability to monitor for change. Although

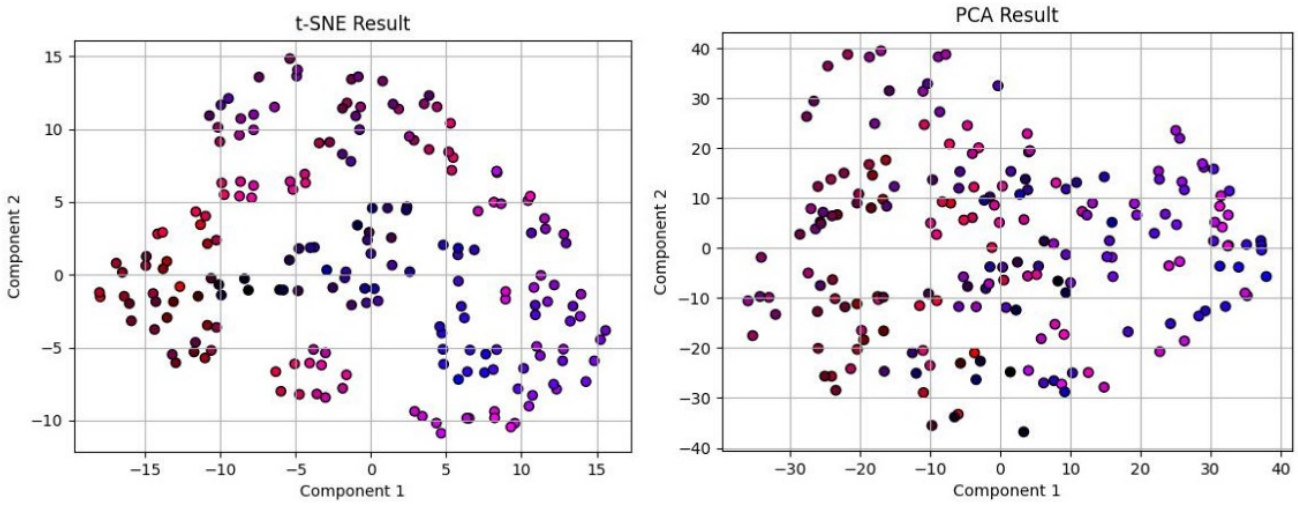

Fig. 6. Visualizing location-variant object embeddings.

the idea is now clear, monitoring these embeddings in any way becomes a complex task due to their high dimensionality.

This paper attempts to tackle this problem by applying two dimensionality reduction techniques, which are used to project the embeddings into a 2-dimensional space, and these lower dimensional embeddings are visualized as a 2d Scatter plot as shown below in Fig.7.

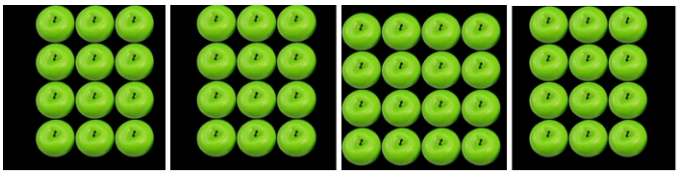

Fig. 7. custom exclusively location-variant data.

1) PCA: : PCA is a worthy dimensionality reduction candidate for this problem because we aim to ascertain how and where the location-specific information exists and changes among different embeddings.

More specifically, the experiment is engineered such that the maximum variation in the data corresponds to localization information since semantic information should be identical for duplicate crops of objects. So essentially, we need to find which component of the embedding most of the variation lies in, and this component of the embedding must ideally be location-specific. Finding the components with maximum variation is what PCA does by design, and hence we have selected his dimensionality reduction technique.

The right side plot in Fig.7 is the result of plotting the projections of the embeddings onto the first 2 principal components. the brightness of the red color is determined by the magnitude of the x value of the top left corner of the bounding box (of the object that produced that particular embedding.). Similarly, the brightness of the blue colour corresponds to the y value of the top left corner of the bounding box.

From Fig.7., we that the dots are redder in higher values of the Y axis of the plot, and bluer when the X axis value is higher., this indicates that the 1st component explains the variance in the y coordinate, and the 2nd component encompasses the variance of the x coordinate of the top left corner of the bounding box.

2) t-SNE: t-SNE, as introduced in citeb6 is a manifold learning method that excels at retaining neighbor structure over a manifold. Whereas, our task requires us to study the variation across the entire possible range of values (0 - image dimension). This makes t-sne a less ideal choice for dimensionality reduction for this task's requirements.

Nevertheless, This visualization was done out of mere curiosity. In the t-SNE plot from Fig.7, we find that dots with similar colors are starting to group. Even though translation was attempted uniformly, in all locations of the image, occluded instances were not used, which means, the edges of the images had fewer object instances than regions in the center. This non-uniformity, along with the dimensional constraints of the objects, allowed t-SNE to enlarge the inherent distances of the objects in the embedding space, resulting in groups as noticed in the t-SNE plot of Fig.7.

V. Conclusion

The focus of this project was to explore the structure of objects and their organization in CNNs' embedding spaces. First crops of objects from images, were obtained using zero-shot prompt-based detection and models, and these

crops were fed into the ResNet50 model to extract the object-specific embeddings. Next, using object detection models, crops of an embedding based on object locations in the image governed by their bounding boxes were used as object-specific embeddings. The structure of the 2 embedding spaces was analyzed by studying the neighbors obtained using the KNN algorithm. Each of the 2 embedding spaces had shown certain interesting behaviors, and these were discussed. Finally, A short quest to understand how and where location-specific information is encoded was undergone, leading to a verification of its presence by visualization of a lower dimensional map of the embedding space. Overall, the many experiments conducted led to a better understanding of how embedding spaces work, how powerful they can be, and how difficult they can be to navigate.

REFERENCES

[1] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in Proceedings of the European Conference on Computer Vision (ECCV), vol. 8689, 2014, ISBN: 978-3-319-10589-5.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2016.

[3] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263–7271

[4] S. Bell and K. Bala, "Learning visual similarity for product design with convolutional neural networks," ACM Trans. Graph., vol. 34, no. 4, Art. no. 98, Aug. 2015, doi: 10.1145/2766959.

[5] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, et al., "Grounded SAM: Assembling Open-World Models for Diverse Visual Tasks," arXiv preprint arXiv:2401.14159, 2024.

[6] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," Journal of Machine Learning Research, vol. 9, pp. 2579-2605, Nov. 2008.