Contrastive Learning

In the last pre-read, we understood **Transfer Learning**, where we can use a pre-trained model to solve the problem at hand. But there can be many instances where we might not find such a pre-trained model for the problem at hand. In that case, we need to gather labeled data, however, in the real world, that may also not always be easy because it can be costly and resource-heavy to acquire high volumes of labeled data.

In comparison to labeled data, unlabeled data is more cost-effective and can be found in ample amounts. We can solve the problem of not being able to feasibly acquire huge labeled datasets, if only we had a way to label the unlabeled images.

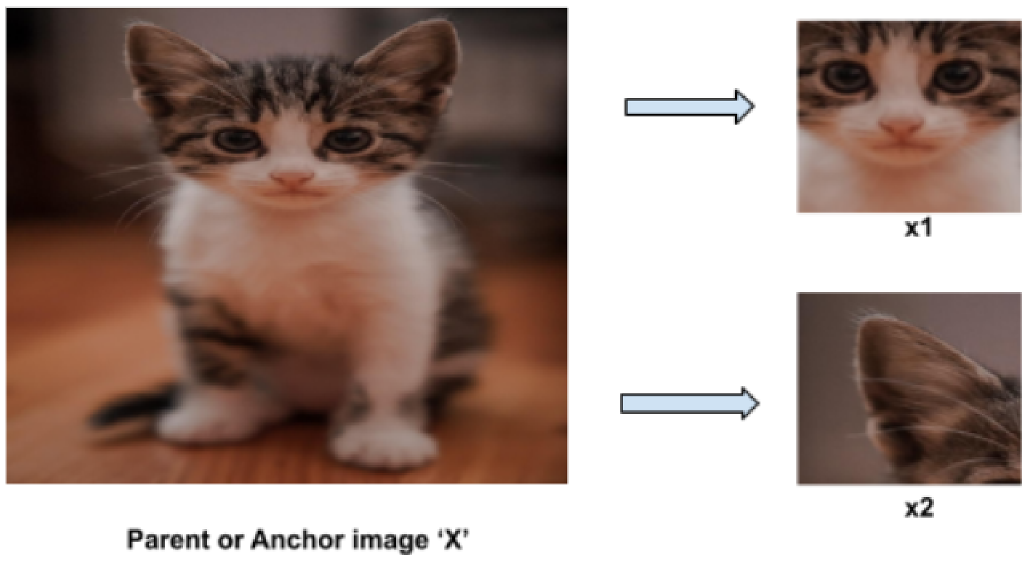
This is where **Contrastive Learning** comes into play.

Contrastive Learning is a deep learning technique that tries to learn similarities within the dataset by looking at the images and their various parts.

Let's try to understand how it works:

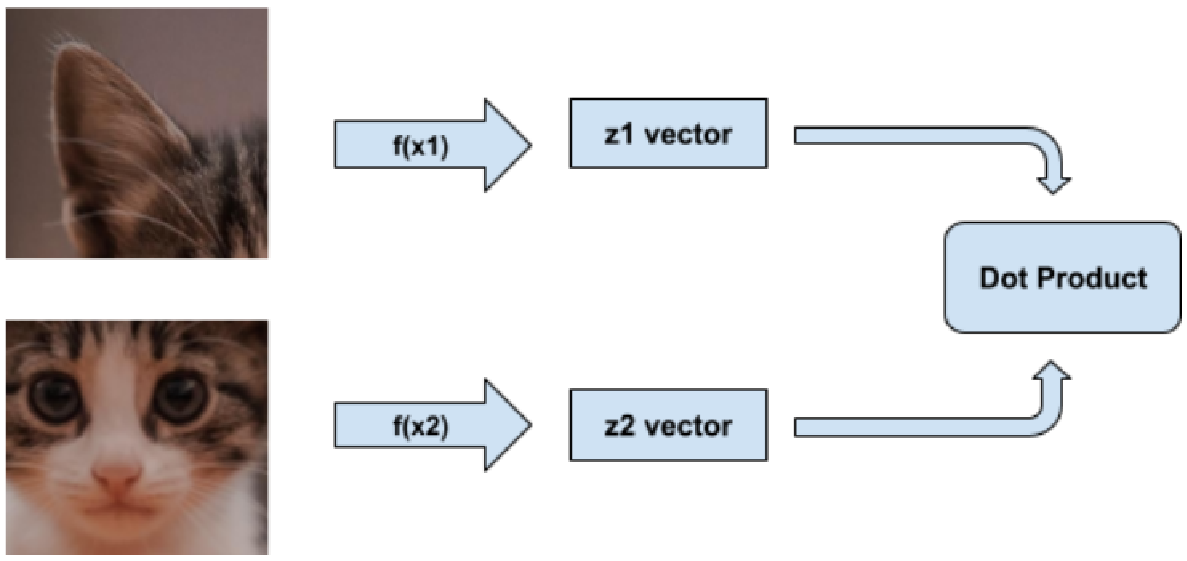
Suppose, we have an image 'X'. Now from 'X', we take two images. They can be smaller cropped portions of 'X', or they could be the augmented versions of 'X'. Let's call them 'x1' and 'x2', respectively.

Since 'x1' and 'x2' both belong to the same parent image X, these are **positive samples**.



Now, since both 'x1' and 'x2' are taken from the same image, we would want our Deep Learning model to predict that they are similar and in the process, learn an image representation.

So, 'x1' and 'x2' go through the function f() and they each produce a 1-D vector, 'z1' and 'z2' as shown in the image below.



Since these images came from the same image, the 1-D vectors 'z1' and 'z2' should be highly similar.

Now, **how do we check for similarity?**

We take the scaled dot product of these two 1-D vectors (dot product divided by the product of magnitudes of each vector), which lies between 0 and 1 (we ignore the sign in case of a negative similarity). We would expect the scaled dot product to be closer to 1 for positive samples. If the value is much lesser than 1, then the difference between 1 and the value we have received is treated as the error and we optimize the function f() accordingly.

Similar to these positive samples, we can have negative samples as well. For example, if 'x1' and 'x2' belonged to different parent images, they are called **negative samples**. In this case, we would expect the similarity score to be zero. And if it is not zero, the difference is treated as an error.

< Previous

Next >