Embedding Shift Dissection on CLIP: Effects of Augmentations on VLM's Representation Learning

Supplementary Material

A. Appendix

We present additional details about our experiment, results and visualizations on the appendix section.

A.1. Hyperparameters and Metrics Details

This section contains the explanation of each variables used on the methodology figure on Fig. 2. The custom metrics section contains metrics that are commonly used by multiple algorithms and research works in recent academia.

A.1.1. SciPy Functions

Cosine Similarity and L2 distance functions were implemented on numpy but are mentioned in this section as they closely align with SciPy's available implementations. Rest of the metrics like fcluster (the dendrogram) the pdist and squareform were used directly from SciPy without any additional modifications.

$$C = \texttt{fcluster}(Z, t, \texttt{criterion} =' \texttt{distance}') \tag{1}$$

where Z is the linkage matrix and t is the distance threshold.

$$D = \texttt{pdist}(X, \texttt{metric} = m) \tag{2}$$

where X is an $n \times m$ matrix and m is the distance metric.

$$S = \texttt{squareform}(D) \tag{3}$$

Converts between condensed and square distance matrices.

$$\text{sim}(u, v) = \frac{u \cdot v}{\|u\|\|v\|} \tag{4}$$

Cosine similarity between vectors u and v.

$$d(u, v) = \sqrt{\sum_{i=1}^{n}(u_i - v_i)^2} \tag{5}$$

L2 distance (Euclidean) between vectors u and v.

A.1.2. Custom Metrics

Our inspiration for these metrics were both derived form previous works [3, 6, 7] as well as recent industry use of such metrics.

$$\text{attn}_{\text{sim}} = \frac{1}{1 + \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}(A_{ij} - A'_{ij})^2} \tag{6}$$

where A and A' are original and augmented attention maps of size $m \times n$.

$$\text{patch}_{\text{sim}} = \frac{1}{16}\sum_{k=1}^{16}\frac{1}{1 + \frac{\text{MSE}_k}{255^2}} \tag{7}$$

where for each patch:

$$\text{MSE}_k = \frac{1}{whc}\sum_{i=1}^{w}\sum_{j=1}^{h}\sum_{c=1}^{3}(P_{ijk} - P'_{ijk})^2 \tag{8}$$

$$\text{edge}_{\text{sim}} = 1 - \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}|E_{ij} - E'_{ij}| \tag{9}$$

where edge maps are computed as:

$$E = \frac{|G_x(I)| + |G_y(I)|}{\max(E)}, \tag{10}$$

$$E' = \frac{|G_x(I')| + |G_y(I')|}{\max(E')} \tag{11}$$

$$\text{detail}_{\text{sim}} = \frac{1}{N}\sum_{k=1}^{N}\exp\left(-\left|\log\left(\frac{\sigma(P'_k)}{\sigma(P_k)}\right)\right|\right) \tag{12}$$

where:

- P_k and P'_k are corresponding patches
- σ is the standard deviation operator
- N is the number of valid patches (excluding uniform ones)

A.1.3. Implementation Notes

- All metrics are averaged over multiple samples; SciPy functions were averaged over all the 13k images whereas custom metrics were averaged over 2,000 unique samples
- Image dimensions: $h \times w$ for height and width
- Grayscale conversion uses $\text{Gray} = 0.299R + 0.587G + 0.114B$
- Gradient operators G_x and G_y are implemented via finite differences
- Patch operations use integer division for grid creation

A.1.4. Augmentations Details

Algorithm 1 presents our algorithm on the hyperparameters related to augmentation of images. We show the entire logic for our current implementation of the code for the custom dataset as well as the various hyperparameters that were passed on to albumentations [2] to create our unique images.

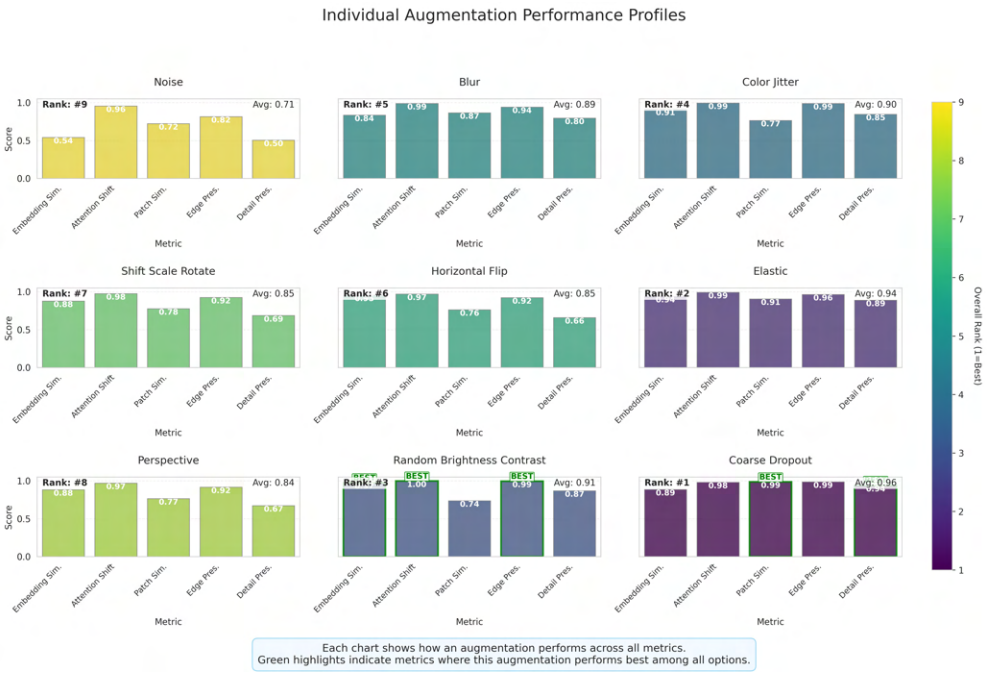


Figure 6. Ranked bar plot for each augmentation profile on performance metrics

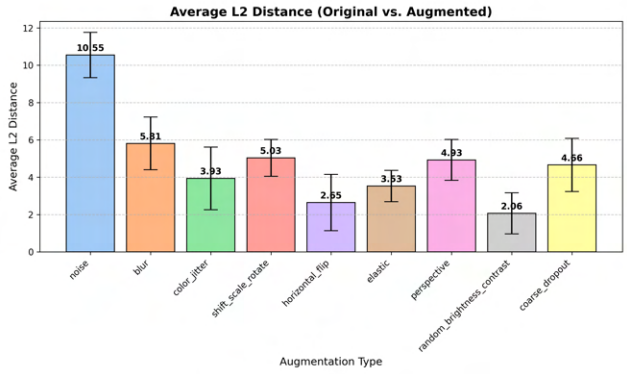


Figure 7. Average L2 distance bar plot for each metric

A.2. Additional Results

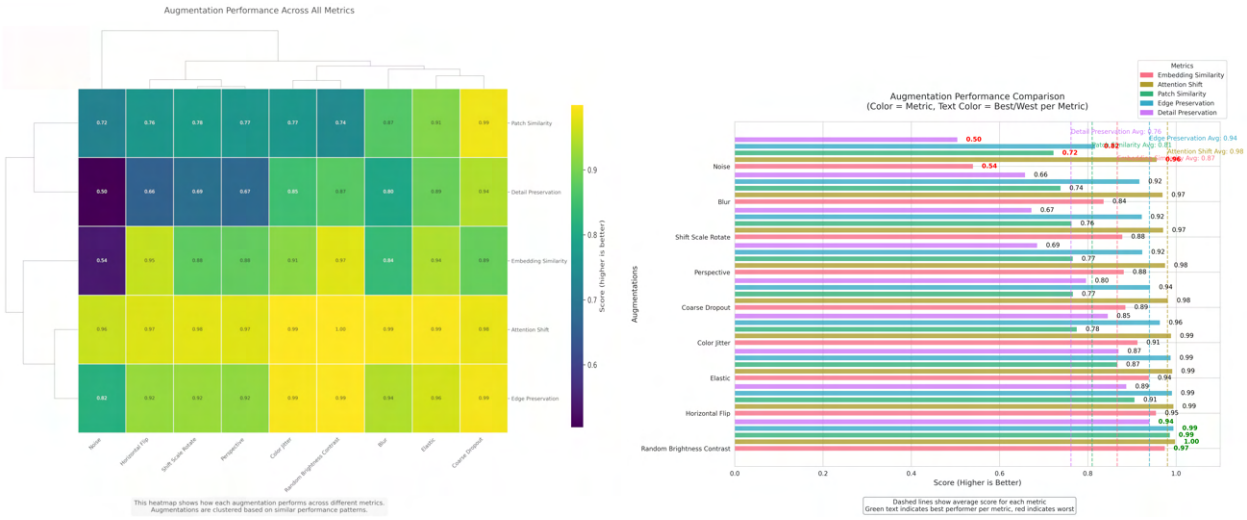
We present a new perspective to the results using different graphs for the quantitative results observed in Sec. 4 and provide more in-depth examples of qualitative results in this section in Figs. 7 to 9. In Fig. 7, we show the average L2 distance of each augmentation's embeddings against the original embeddings. This is a further intuitive explanation of the KDE plot in Fig. 4b. Fig. 6 shows a rank fashion bar plot ranking each of the augmentation based on average performance across all metrics. It provides more visual intuition towards the results observed in Fig. 4c.

Fig. 8a show a combined intuition towards the dendrogram clustering in Fig. 4d combined together with Fig. 3. We also took 50 random samples and evaluated the cosine similarity of each sample's augmented representation with the original representation to check for metrics consistency and report it on Fig. 9. An unsorted version of Fig. 6 that instead highlights the overall average of each metrics is presented on Fig. 8b.

Following pages contain some of the qualitative analysis we have conducted that are an extension of the abstract and visualization for a comprehensive review of the paper Fig. 1 and Fig. 4a.

Algorithm 1 Image Transformation Dataset Processing

procedure IMAGETRANSFORMDATASET($image\_dir$, $transforms\_dict$, $image\_size$)
    Initialize:
    self.image_dir ← Path($image\_dir$)
    self.image_paths ← Collect image paths (**.jpg, **.jpeg, **.png)
    Print dataset size: |self.image_paths|
    Base Transform:
    self.base_transform ← Resize($height = image\_size[0], width = image\_size[1]$)
    if $transforms\_dict = \emptyset$ **then**
        Set default transformations:
        self.transforms_dict ← { (1) GaussNoise($std = (0.44, 0.88), p = 1.0$),
        (2) GaussianBlur($kernel = (3, 7), p = 1.0$),
        (3) ColorJitter($brightness/contrast/saturation/hue = 0.2, p = 1.0$),
        (4) ShiftScaleRotate($shift = 0.0625, scale = 0.1, rotate = 15°, p = 1.0$),
        (5) HorizontalFlip($p = 1.0$), }
        (6) ElasticTransform($\alpha = 30, \sigma = 60, p = 1.0$),
        (7) Perspective($scale = (0.05, 0.1), p = 1.0$),
        (8) RandomBrightnessContrast($limit = 0.2, p = 1.0$),
        (9) CoarseDropout($num\_holes = 6 - 8, size = 16 \times 16, fill =$ random, $p = 1.0$)
    else
        self.transforms_dict ← $transforms\_dict$
    end if
end procedure
function GETITEM(idx)
    $image\_path$ ← self.image_paths[idx]
    $image$ ← ReadRGB($image\_path$)
    $original$ ← ApplyTransform($image$, self.base_transform)
    Initialize result dictionary:
    result ← {"image_path": $image\_path$, "original": $original$}
    for each $(name, transform)$ **in** self.transforms_dict **do**
        $transformed$ ← ApplyTransform($original, transform$)
    end for
    return *result*
end function



(a) Heatmap with dendrogram clustering over evaluation metrics (b) Average augmentation performance comparison on custom metrics

Figure 8. Additional quantitative analysis results

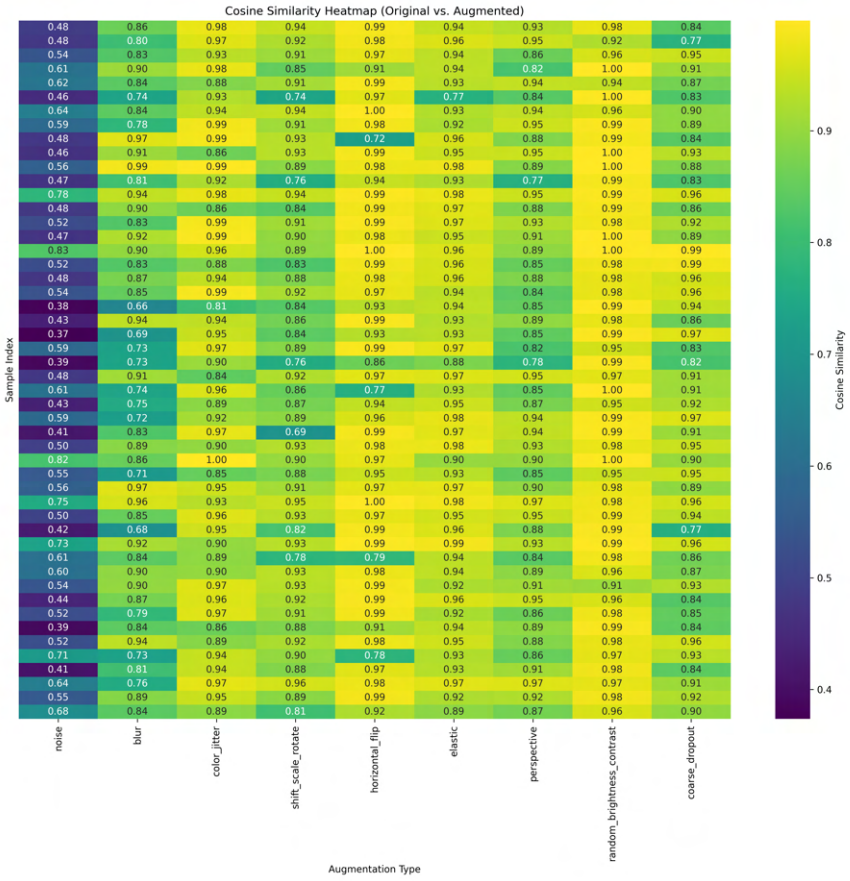


Figure 9. Heatmap of 50 random sample's cosine similarity towards the embedding of original image

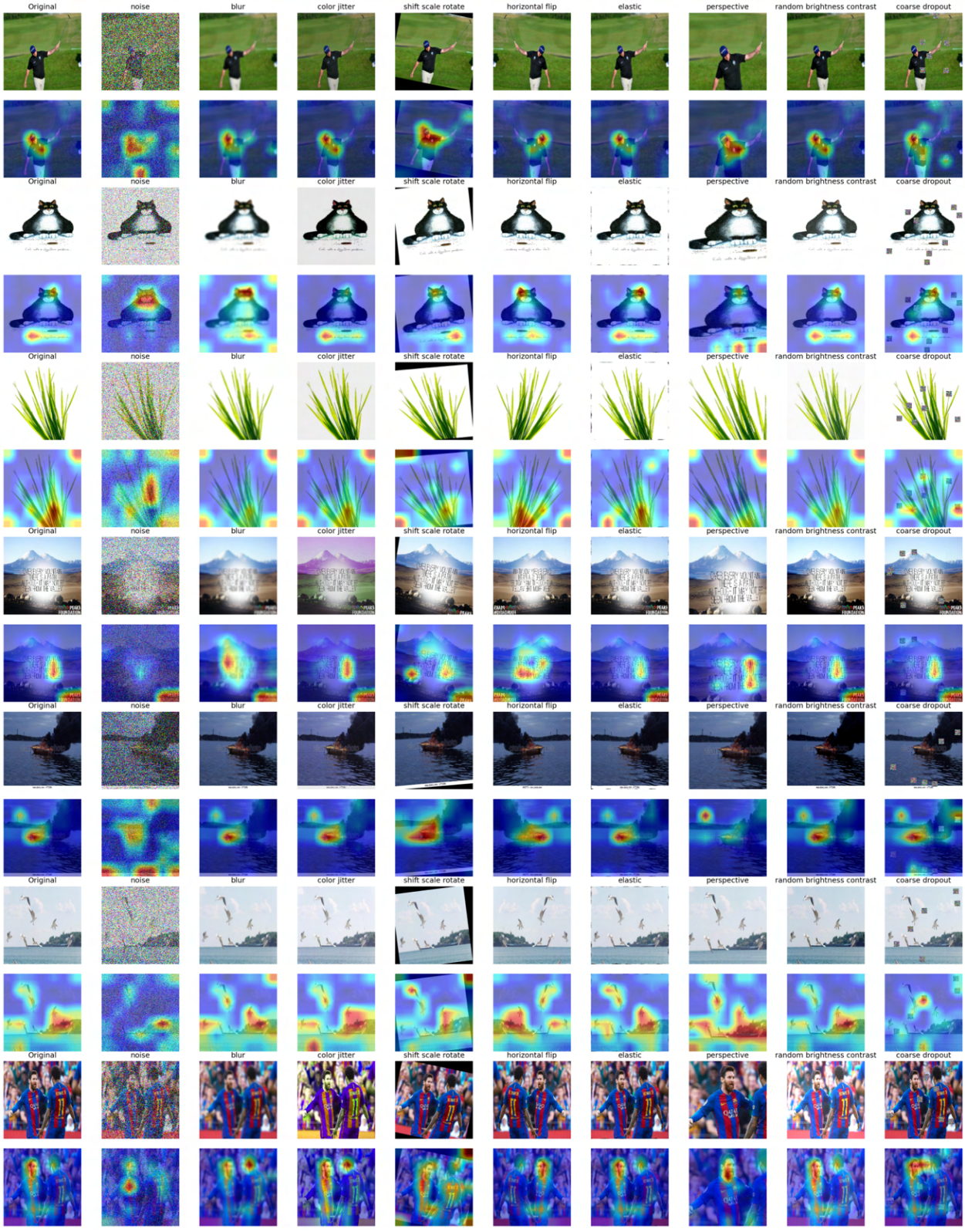


Figure 10. Sample qualitative analysis of attention map for each augmentation types

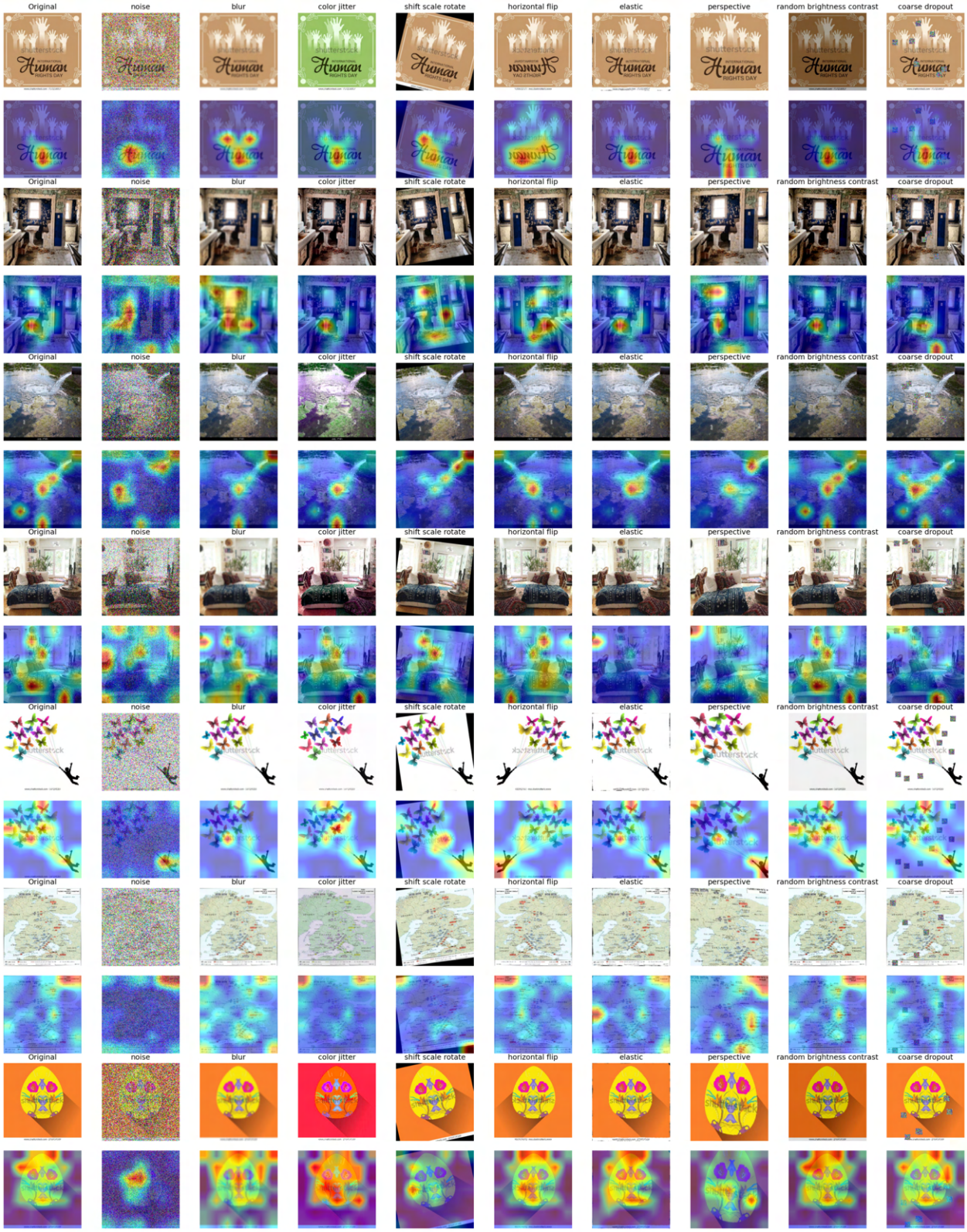


Figure 11. Sample qualitative analysis of attention map for each augmentation types