Understanding Dataset Bias in Medical Imaging: A Case Study on Chest X-rays

Supplementary Material

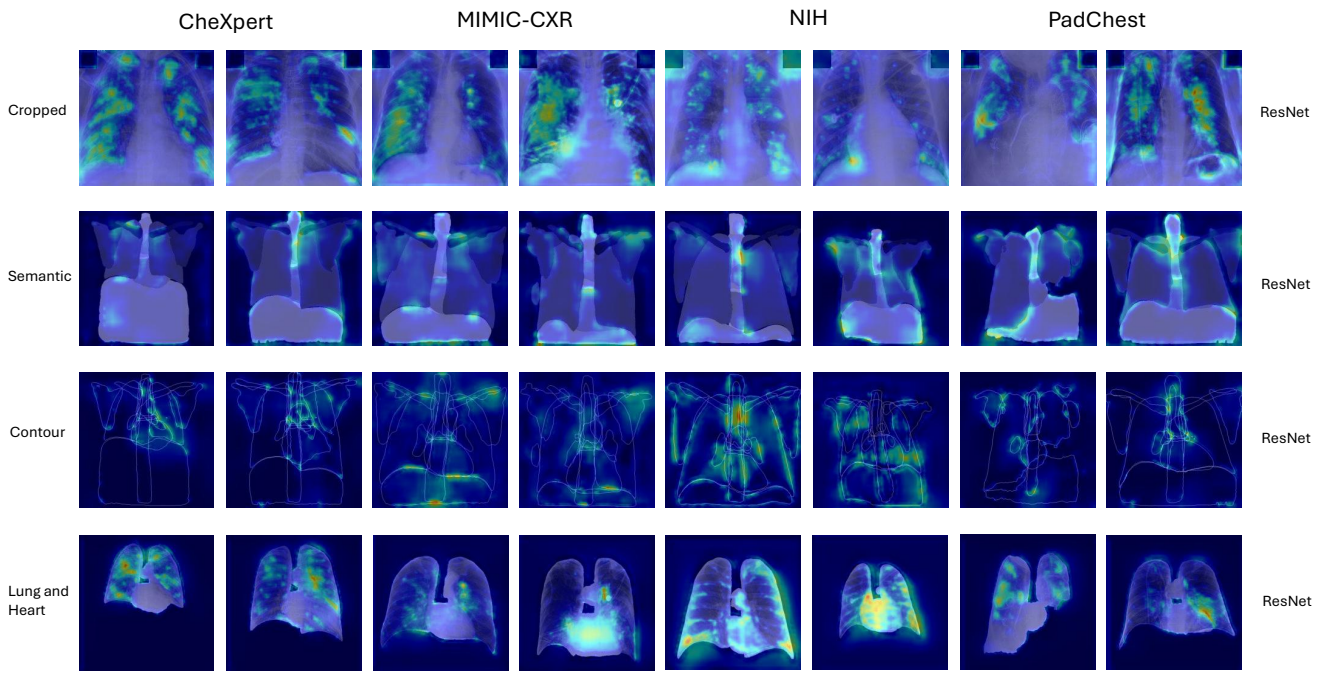

Figure 1. Likewise, as in the first heatmap figure, we pass the same images through their corresponding best-performing model and generate heatmaps based on all layers to visualise the model's prediction process.

1. Average Heatmaps

In the previous heatmap figure, the Grad-CAM was generated using only the final activation layer. As a comparison, we now generate heatmaps by averaging the Grad-CAMs computed across all model layers of the best-performing model. Looking at Figure 1, while the heatmaps for the cropped and LH images remain relatively similar, the semantic and contour images provide new insights. These reveal a broader range of regions the model attends to, and the attention patterns appear less random than those based on the final layer alone. The model seems to be analysing various anatomical structures to determine the dataset of origin. This suggests a form of semantic-level analysis, similar to what we computed, where the model approximates organ structures and assesses which dataset they most likely correspond to.

2. Transformations

We utilise the **MONAI library** to implement 13 carefully selected augmentation transformations designed to enhance model robustness through controlled variations in pixel intensity and texture. Our experiments evaluate these transformations using two distinct application probabilities: $P = 0.2$ (conservative) and $P = 0.5$ (aggressive).

Augmentation Transformations

The complete set of transformations includes:

Intensity Modifications

- **Gaussian Noise**
  `RandGaussianNoise` (prob = P)
- **Intensity Shifting**
  `RandShiftIntensity` (offset = 0.1, prob = 0.5)
  `RandStdShiftIntensity` (factor = 0.1, prob = 0.5)
- **Intensity Scaling**
  `RandScaleIntensity` (factor = 0.1, prob = 0.5)
  `RandScaleIntensityFixedMean` (factor = 0.1,

prob = 0.5)

- **Contrast Adjustment**
  `RandAdjustContrast` (prob = 0.5)

Spatial & Texture Modifications

- **Smoothing Filters**
  `SavitzkyGolaySmooth` (window_length = 5, order = 2, prob = 0.5)
  `RandGaussianSmooth` (σ = 1.0, prob = 0.5)
  `MedianSmooth` (radius = 1, prob = 0.5)
- **Sharpening**
  `RandGaussianSharpen` (prob = 0.5)
- **Non-linear Transforms**
  `RandHistogramShift` (control_points = 10, prob = 0.5)

Structural Perturbations

- **Dropout & Shuffling**
  `RandCoarseDropout` (holes = 5, size = (32,32), prob = 0.5)
  `RandCoarseShuffle` (holes = 5, size = (32,32), max_holes = 10, prob = 0.5)

We aim to preserve the existing transform pipeline used during training and simply insert a MONAI transform within it. Since MONAI expects input as NumPy arrays with a single channel, we add a custom transform after resizing the PIL image to convert it to a NumPy array and append a channel dimension. After applying the MONAI transforms, we convert the output (a MONAI MetaTensor) back to a NumPy array and apply another custom transform to remove the channel dimension. This ensures compatibility while keeping the rest of the transformation pipeline unchanged.