# Image Pre-processing Pipeline

The following document outlines the exact sequence of pre-processing transformations applied to every image in both the training and validation datasets. The pipeline is implemented using torchvision.transforms and is designed to prepare images for a deep learning model.

## 1. Image Resizing

- **Transformation**: transforms.Resize((112, 112))
- **Description**: Every input image is resized to a fixed resolution of **112×112 pixels**.
- **Method**: This transformation does not preserve the original aspect ratio; it **squashes** or stretches the image to fit the target dimensions. The default interpolation method used for PIL images is **bilinear interpolation**.

## 2. Conversion to PyTorch Tensor

- **Transformation**: transforms.ToTensor()
- **Description**: The resized image, which is typically in a format like a PIL Image or NumPy array, is converted into a PyTorch tensor.
- **Key Changes**:
  - **Dimension Ordering**: The tensor's dimensions are reordered from Height × Width × Channels (H×W×C) to **Channels × Height × Width (C×H×W)**, which is the standard format for PyTorch models.
  - **Data Scaling**: Pixel intensity values are scaled from the original range of [0, 255] to a floating-point range of [0.0, 1.0].
  - **Color Channel Assumption**: This transform assumes the input image is in the RGB color space.

## 3. Channel-wise Normalization

- **Transformation**: transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
- **Description**: Each channel of the image tensor is normalized using the mean and standard deviation statistics from the ImageNet dataset.
- Formula: The normalization is applied element-wise for each channel $c \in \{R,G,B\}$ according to the formula:

  $xc' = \sigma c x c - \mu c$

  Where:
  - $\mu = (0.485, 0.456, 0.406)$ are the mean values for the R, G, and B channels.
  - $\sigma = (0.229, 0.224, 0.225)$ are the standard deviation values for the R, G, and B channels.
- **Purpose**: This step centers the data around zero with a standard deviation of approximately one, which helps stabilize and accelerate the model's training process.

---

## Summary and Considerations

The pre-processing pipeline is deterministic and identical for both the training and validation sets. Notably, **no data augmentation** techniques (e.g., random flips, rotations, crops, or color jittering) are employed.

### Potential Improvements & Fixes

1. **Aspect Ratio Distortion**: If preserving the geometry of objects in the images is important (e.g., distinguishing between circles and ovals), the current resizing method can be problematic. A better approach would be to resize the smaller edge to a certain size and then take a center crop, for example: transforms.Resize(128) followed by transforms.CenterCrop(112).
2. **Handling Grayscale Images**: The current pipeline will fail if it encounters a grayscale image because it expects 3-channel (RGB) input. To handle this, one could add transforms.Grayscale(num_output_channels=3) before transforms.ToTensor() to convert any single-channel images to 3-channel.
3. **Dataset-Specific Statistics**: While using ImageNet statistics is a common and effective practice, model performance can often be slightly improved by calculating and using the **mean and standard deviation specific to your own dataset**. This ensures the normalization is perfectly tailored to the distribution of your data.