LEGO BRICKS DATASET PREPARATION

Sourse: https://zenodo.org/records/10731976

This dataset introduces a simple, low-resolution (640x640) object detection dataset for manufacturing, specifically focused on identifying minor surface defects in plastic Lego bricks. It is designed to address the need for accessible, less complex datasets, that can help small enterprises to adopt object detection for quality control.

The dataset contains about 1,500 annotated images, with two quality labels (defect/no defect). Originally it was tested using YOLOv5 to demonstrate the feasibility of accurate defect detection with limited, low-resolution data.

I adjusted it to my personal scope: **Object detection for surface defect classification images of plastic bricks.**

The core problem:

- Each image can have multiple bricks.
- Some bricks in the image may be defective, some may not.

I choose to classify the whole image as "defect" or "no_defect".

My Approache:

- Image is 'defect' if ANY brick is defected.
- If all objects are non-defective, label as 'no_defect'.

we need to achieve: train/defect/, train/no_defect/, valid/defect/, valid/no_defect/

Note: the old images/ and labels/ folders will be deleted.

In [18]:
```
import os
import shutil
```

In [20]:
```
def get_class_from_label(label_path):
    """
    Reads a YOLO label file and determines the class for the image.
    If any line in the label file starts with '0', it's a 'defect'.
    Otherwise, it's 'no_defect'.
    """
    with open(label_path, 'r') as f:
        for line in f:
            if line.strip() and line.strip().split()[0] == '0':
                return 'defect'
    return 'no_defect'

def organize(split):
    """
    For a given split ('train' or 'valid'),
    - Reads each image in images/,
    - Determines its class from the corresponding label file in labels/,
    - Copies the image to the appropriate class folder (defect/no_defect)
    """
    images_dir = os.path.join(split, 'images')
    labels_dir = os.path.join(split, 'labels')
    for img_name in os.listdir(images_dir):
        # Only process image files
        if not img_name.lower().endswith(('.jpg', '.jpeg', '.png')):
            continue
        label_name = os.path.splitext(img_name)[0] + '.txt'
        label_path = os.path.join(labels_dir, label_name)
        if not os.path.exists(label_path):
            print(f"Warning: No label for {img_name}, skipping.")
            continue
        class_name = get_class_from_label(label_path)
        dest_dir = os.path.join(split, class_name)
        os.makedirs(dest_dir, exist_ok=True)
        shutil.copy2(os.path.join(images_dir, img_name), os.path.join(des
```

In [22]:
```
# Organize both train and valid splits

for split in ['train', 'valid', 'test']:
    organize(split)
print("Done! The data is now organized for classification.")

for split in ['train', 'valid', 'test']:
    for sub in ['images', 'labels']:
        shutil.rmtree(f'{split}/{sub}', ignore_errors=True)
print("Old images/ and labels/ folders removed.")
```

```
Done! The data is now organized for classification.
Old images/ and labels/ folders removed.
```

In [ ]: