Operation FractureScope: Dataset Preparation and Model Architecture

Sri Ram

May 7, 2025

1 Proposed Dataset Preparation and Model Architecture

1.1 Dataset Collection and Organization

To construct a comprehensive multi-modal surface defect dataset, we aggregated samples from three publicly available datasets:

- **NEU Surface Defect Dataset (NEU-DET)** — Provides various steel defect images.
- **SDNET2018** — Contains concrete crack images from decks, pavements, and walls.
- **Severstal Steel Defect Dataset** — Includes steel surface defect images with annotated classes.

We focused on curating three defect classes:

1. **Crack**: Sourced from SDNET's cracked subfolders (Decks, Pavements, Walls).
2. **Corrosion**: Extracted from NEU dataset's *pitted_surface* and *rolled-in_scale* categories.
3. **Scratch**: Gathered from NEU dataset's *scratches* category and Severstal ClassId=4 (Scratch).

The collected dataset was organized into class-specific folders. The final image counts per class are summarized in Table 1.

Table 1: Summary of collected dataset images.

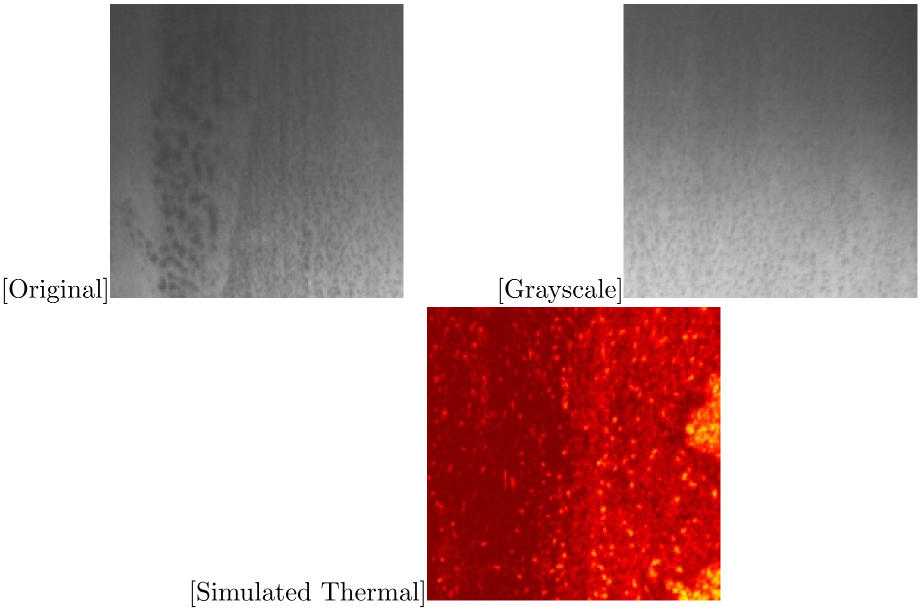
Class	**Number of Images**
Crack	8,487
Corrosion	483
Scratch	243

1.2 Multi-modal Image Generation

To simulate multi-modal inputs akin to drone-based inspections, three modalities were generated for each image:

1. **Original** (RGB image)
2. **Grayscale** (single-channel intensity image)
3. **Simulated Thermal** (pseudo-color image using *COLORMAP_JET*)

An example visualization of generated modalities is shown in Figure 1.


[Original]

[Grayscale]

[Simulated Thermal]

Figure 1: Example of generated modalities: (a) Original, (b) Grayscale, (c) Simulated Thermal.

1.3 Train-Validation Split

For robust model evaluation, an 80:20 stratified train-validation split was applied to each class and modality. The resulting dataset sizes are shown in Table 2.

Table 2: Train and Validation sample distribution.

Split	Train Samples	Validation Samples
Overall	27,384	13,566

1.4 Model Architecture: FractureNet

A custom deep learning classifier named **FractureNet** was proposed. The architecture is based on a ResNet18 backbone, with modifications:

- The first convolutional layer was modified to accept 3-channel images (standard RGB or modality composites).
- The final fully connected layer was modified to output logits for 3 classes (*crack, corrosion, scratch*).

1.5 Dataset Loader and Transformations

To prepare the dataset for training, standard preprocessing and data augmentation steps were applied:

- Images were resized to 224 × 224 resolution.
- Normalization was applied using ImageNet mean and standard deviation.

The dataset loader pipeline was implemented using PyTorch's `Dataset` and `DataLoader` classes. The core implementation code is shown in Listing 1.

Listing 1: FractureDataset class implementation

```
class FractureDataset(Dataset):
    def __init__(self, data_dir, classes, modalities, transform=None):
        self.data_dir = data_dir
        self.classes = classes
        self.modalities = modalities
        self.transform = transform
        self.image_paths = []
        self.labels = []

        for idx, cls in enumerate(classes):
            cls_dir = os.path.join(data_dir, cls)
            for modality in modalities:
                modality_dir = os.path.join(cls_dir, modality)
                for fname in os.listdir(modality_dir):
                    self.image_paths.append(os.path.join(modality_dir,
                        fname))
                    self.labels.append(idx)

    def __len__(self):
        return len(self.image_paths)

    def __getitem__(self, idx):
        img_path = self.image_paths[idx]
        img = Image.open(img_path).convert(’RGB’)

        if self.transform:
            img = self.transform(img)

        label = self.labels[idx]
        return img, label
```

The data loaders were configured with a batch size of 16 and appropriate shuffling for training:

Listing 2: Train and Validation DataLoader setup

```
train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=16, shuffle=False)
```