Graphical user interface, application

Description automatically generated

1. **Data Collection**
   1. 80 images of burst pipe are collected.
   2. All sample are collected from the internet.
   3. The samples are put together in one directory.

Graphical user interface, text

Description automatically generated

**Figure 1: All samples are put together in one directory**

A picture containing metalware, rock, catch

Description automatically generatedA picture containing building material, stone

Description automatically generated

A close up of a piece of wood

Description automatically generated with low confidenceA close up of a piece of wood with holes in it

Description automatically generated with low confidence

**Figure 2: 4 of 80 samples of burst pipe image**

1. **Data Augmentation**
   1. All datasets are augmented to increase the variety of targeted object.
   2. The higher amount of sample, the higher accuracy of detection.
   3. There are five method of augmentation which are resized, rotation, contrast, and salt & pepper.
   4. Dataset after augmentation process is increase from 80 to 720 samples.

**Table 1: Augmentation method**

|  |  |
| --- | --- |
| **Method** | **Result** |
| Resized  (1000x1000) | A close-up of a pipe  Description automatically generated with low confidence |
| Rotation  (45 and 315 degree) |  |
| Contrast |  |
| Salt & Pepper |  |

1. **Data Annotation**
   1. Annotate all the sample to get the region boundary of targeted object.
   2. Name the annotation as ‘burst’.

A screenshot of a computer

Description automatically generated with low confidence

**Figure 3: Annotate sample as ‘burst’ using labelimg**

1. **Data Splitting**
   1. Split all samples into three categories which are train, valid, and test with ratio 70:20:10.
   2. Train sample is used to teach the model to recognize the targeted object.
   3. Valid sample is used to avoid overfitting during the training process.
   4. Test sample is used to evaluate the model that has been trained.



**Figure 4: Data splitting into three categories**

1. **Data Training**
   1. Train the dataset to reduce the validation loss as much as it can.
   2. The training process took 28 epochs (~6 hours).
   3. After finish, the model that has been trained is convert into tflite format to let the model compatible with smartphone environment.
2. **Model Deployment**
   1. The model that has been trained is deployed in the application using android studio.
   2. The application read the model so it can be used to detect a burst pipe using smartphone through application.
3. **Data Testing**
   1. After deployed the model in the application, test the accuracy model by detecting the burst pipe using test sample during splitting the dataset.
   2. The accuracy can be evaluated by using this formula:



**Figure 5: Accuracy formula for evaluation model**

**Table 2: Testing result**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Sample** | **Test detection** | **Detection result** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✕** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✕** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✕** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✕** |
|  |  |  | **✕** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✕** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✕** |
|  |  |  | **✓** |
|  |  |  | **✕** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✓** |
|  |  |  | **✕** |
|  |  |  | **✓** |

Total ✓: 60/69

Total **✕:** 9/69

**Region Detection Accuracy = 60/69 x 100 = 86.96%**

**Reference**

Train Model

<https://colab.research.google.com/github/googlecodelabs/odml-pathways/blob/main/object-detection/codelab2/python/Train_a_salad_detector_with_TFLite_Model_Maker.ipynb>

Dataset Augmentation

<https://roboflow.com/>

Application Deployment

<https://github.com/tensorflow/examples/tree/master/lite/examples/object_detection/android>