**BAB IV**

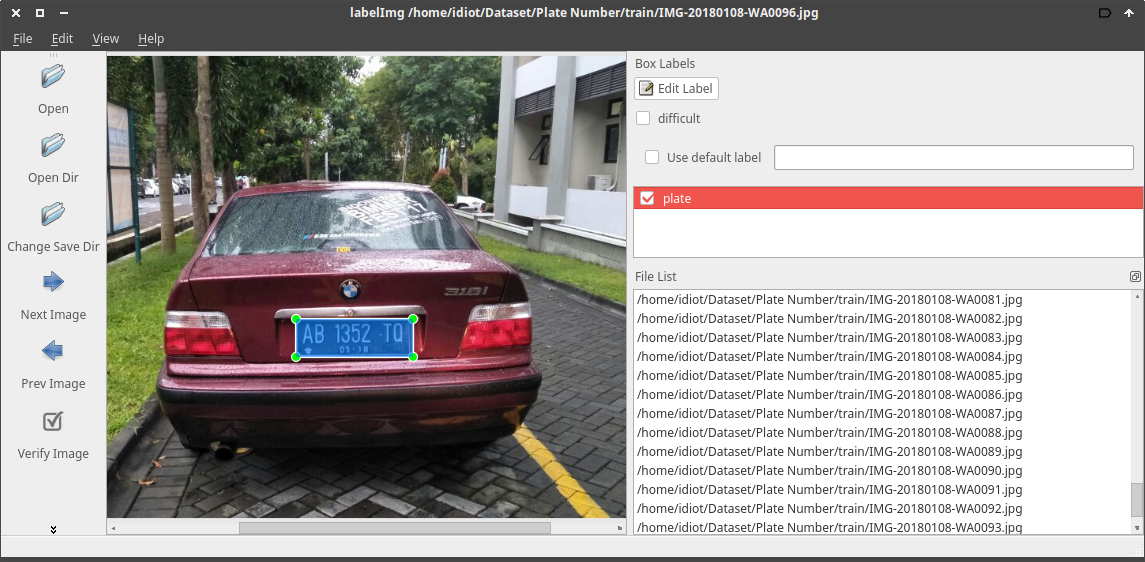
**IMPLEMENTASI DAN PEMBAHASAN SISTEM**

* 1. **Implementasi**

Implementasi sistem merupakan tahapan dari perancangan sistem yang telah dibuat, serta menguji dan memulai penggunaan sistem. Berikut adalah cuplikan program dan alur pembuatan yang merupakan inti dari sistem.

* + 1. **Pelabelan Gambar**

Pelabelan gambar adalah tahap awal dimana dataset inputan diberikan label atau pengenal (tanda) dengan tujuan untuk menyimpan informasi gambar yang selanjutnya disimpan dalam berkas dengan format XML :



* + 1. **Konversi *Datasets***

Setelah proses pelabelan gambar dengan ouput berupa file perlu adanya konversi ke *TensorFlow Record* file yang digunakan untuk *feeding* data untuk proses training, berikut cuplikan kode pembuatan TFRecord :

|  |
| --- |
| from \_\_future\_\_ import division  from \_\_future\_\_ import print\_function  from \_\_future\_\_ import absolute\_import  import os  import io  import pandas as pd  import tensorflow as tf  from PIL import Image  from object\_detection.utils import dataset\_util  from collections import namedtuple, OrderedDict  flags = tf.app.flags  flags.DEFINE\_string('csv\_input', '', 'Path to the CSV input')  flags.DEFINE\_string('output\_path', '', 'Path to output TFRecord')  FLAGS = flags.FLAGS  def class\_text\_to\_int(row\_label):  if row\_label == 'plate':  return 1  else:  None  def split(df, group):  data = namedtuple('data', ['filename', 'object'])  gb = df.groupby(group)  return [data(filename, gb.get\_group(x)) for filename, x in zip(gb.groups.keys(), gb.groups)]  def create\_tf\_example(group, path):  with tf.gfile.GFile(os.path.join(path, '{}'.format(group.filename)), 'rb') as fid:  encoded\_jpg = fid.read()  encoded\_jpg\_io = io.BytesIO(encoded\_jpg)  image = Image.open(encoded\_jpg\_io)  width, height = image.size  filename = group.filename.encode('utf8')  image\_format = b'jpg'  xmins = []  xmaxs = []  ymins = []  ymaxs = []  classes\_text = []  classes = []  for index, row in group.object.iterrows():  xmins.append(row['xmin'] / width)  xmaxs.append(row['xmax'] / width)  ymins.append(row['ymin'] / height)  ymaxs.append(row['ymax'] / height)  classes\_text.append(row['class'].encode('utf8'))  classes.append(class\_text\_to\_int(row['class']))  tf\_example = tf.train.Example(features=tf.train.Features(feature={  'image/height': dataset\_util.int64\_feature(height),  'image/width': dataset\_util.int64\_feature(width),  'image/filename': dataset\_util.bytes\_feature(filename),  'image/source\_id': dataset\_util.bytes\_feature(filename),  'image/encoded': dataset\_util.bytes\_feature(encoded\_jpg),  'image/format': dataset\_util.bytes\_feature(image\_format),  'image/object/bbox/xmin': dataset\_util.float\_list\_feature(xmins),  'image/object/bbox/xmax': dataset\_util.float\_list\_feature(xmaxs),  'image/object/bbox/ymin': dataset\_util.float\_list\_feature(ymins),  'image/object/bbox/ymax': dataset\_util.float\_list\_feature(ymaxs),  'image/object/class/text': dataset\_util.bytes\_list\_feature(classes\_text),  'image/object/class/label': dataset\_util.int64\_list\_feature(classes),  }))  return tf\_example  def main(\_):  writer = tf.python\_io.TFRecordWriter(FLAGS.output\_path)  path = os.path.join(os.getcwd(), 'images')  examples = pd.read\_csv(FLAGS.csv\_input)  grouped = split(examples, 'filename')  for group in grouped:  tf\_example = create\_tf\_example(group, path)  writer.write(tf\_example.SerializeToString())  writer.close()  output\_path = os.path.join(os.getcwd(), FLAGS.output\_path)  print('Successfully created the TFRecords: {}'.format(output\_path))  if \_\_name\_\_ == '\_\_main\_\_':  tf.app.run() |

* + 1. ***Training***
    2. ***Testing***
    3. **Model**
  1. **Pembahasan Sistem**

Pembahasan sistem merupakan hasil implementasi dan uji coba sistem secara fungsional. Berikut ini akan dijabarkan mengenai sistem yang telah dibuat secara fungsional.

* + 1. ***Resizing***
    2. ***Grayscaling***
    3. ***Training***
    4. ***Testing***
    5. **Model**
  1. **Perhitungan Manual Metode *Convolutional Neural Network***
     1. **Pembobotan**
     2. **Aktifasi**
     3. ***Pooling***
     4. **ReLu**
     5. ***Fully-connected Layer***