Optimizing Model with Dynamic Input Shape using Intel® Distribution of OpenVINO™ Toolkit for Implementing Unconstrained Automatic License Plate Recognition

White Paper

September 2022

Authors:

Ramesh Perumal

Lakshmi Talluru

Document number: 746745-1.0

Contents

Tables

Figures

Revision History

Date	Revision	Description
September 2022	1.0	Initial release.

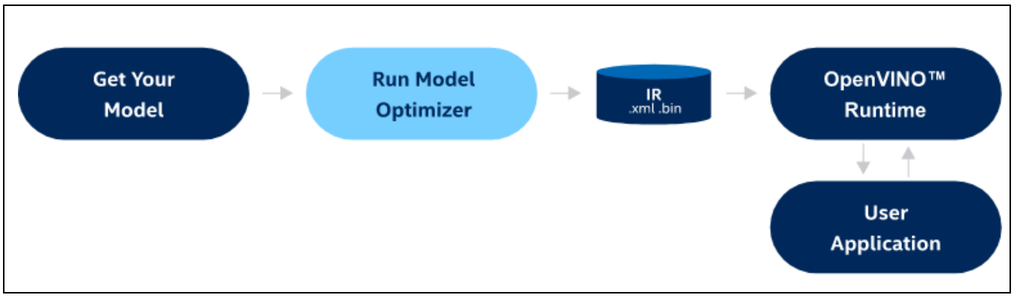
1.0 Introduction

This document introduces the BKM for optimizing and implementing the unconstrained automatic license plate recognition (ALPR) using OpenVINO™ Toolkit 2022.1. Unlike the conventional methods, the unconstrained ALPR can detect distorted license plates (LPs) captured at non-frontal views. It is achieved by resizing the input vehicle image according to its aspect ratio before being fed to the license plate detection model.

However, the detection performance degrades if the input image is resized to the predefined shape as in the case of model optimized with static input shape in the previous OpenVINO™ versions. To overcome this, the dynamic input shape feature introduced in OpenVINO™ 2022.1 enables the optimized model to preserve the undefined input dimensions of the raw model. Due to this, the optimized model could execute the inference on the input images that are being resized at runtime to achieve optimal detection and alignment of distorted LPs.

The Intel® Distribution of OpenVINO™ (Open Visual Inference & Neural Network Optimization) Toolkit enables developers to quickly optimize and deploy the AI workloads with improved performance across the Intel® platforms from edge to cloud. The following figure illustrates the workflow for optimizing and deploying a pre-trained model using Model Optimizer and OpenVINO™ Runtime API.

Figure 1. Workflow for optimizing and deploying a pre-trained model on Intel platform using OpenVINO™



Model Optimizer is a command-line tool that creates an intermediate representation (IR) of the model to facilitate the optimal execution of inference across the Intel® devices (CPU, GPU, VPU). The OpenVINO™ Runtime API contains the hardware-specific plugins for implementing the inference with the optimized model in IR format. The optimized model with dynamic input shape is beneficial when the inference is executed on the image whose input shape is determined only at runtime. To demonstrate its capability, this white paper presents the following:

i. generating the optimized license plate detection model with dynamic input shape using OpenVINO™ 2022.1;
ii. implementing the unconstrained ALPR using the optimized model for detecting the distorted LPs;
iii. comparing the detection performance of the models optimized with static and dynamic input shape using OpenVINO™ 2021.4 and 2022.1.

1.1 Acronyms

Table 1. Acronyms

Term	Description
ALPR	Automatic License Plate Recognition
LP	License Plate
BKM	Best Known Method
OpenVINO™	Open Visual Inference & Neural Network Optimization
IR	Intermediate Representation
WPOD-NET	Warped Planar Object Detection Network
OCR	Optical Character Recognition
BB	Bounding Box

1.2 Reference Documents

Log in to the Resource and Documentation Center (rdc.intel.com) to search and download the document numbers listed in the following table. Contact your Intel field representative for access.

Note: Third-party links are provided as a reference only. Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether the referenced data is accurate.

Table 2. Reference Document

Document	Document No./Location
OpenVINO™	https://software.seek.intel.com/openvino-toolkit
Unconstrained ALPR	https://link.springer.com/chapter/10.1007/978-3-030-01258-8_36
Dynamic Input Shape	https://docs.openvino.ai/latest/openvino_docs_OV_UG_DynamicShapes.html#undefined-dimensions-out-of-the-box
Setting input shape with undefined dimension in Model Optimizer	https://docs.openvino.ai/latest/openvino_docs_MO_DG_prepare_model_convert_model_Converting_Model.html#when-to-specify-input-shape-command-line-parameter

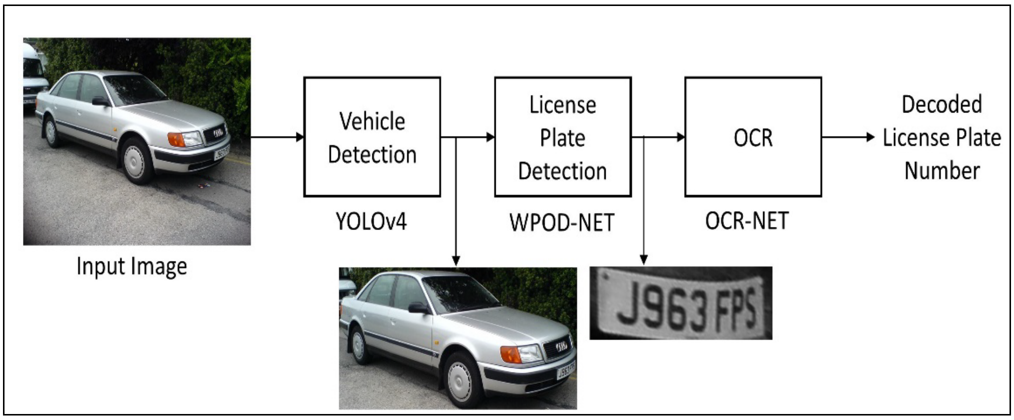
2.0 *Unconstrained ALPR*

2.1 Overview

Most of the existing ALPR methods are limited to detecting the vehicle license plates (LPs) captured at frontal views. Due to this, the detection performance degrades with the distorted LPs in unconstrained scenarios where the vehicles are captured at non-frontal views due to oblique shooting angles. As shown in Figure 2, the ALPR system involves three stages:

1. Vehicle detection,
2. License plate detection, and
3. Optical character recognition (OCR).

Figure 2. Unconstrained ALPR Pipeline



The vehicle detection with YOLOv4 facilitates the detection of multiple vehicles in the input image. The resulting detection outputs are resized based on the vehicle bounding box (BB) dimensions (*refer to Section 2.2 Resizing*) and then fed to the LP detection based on Warped Planar Object Detection Network (WPOD-NET).

The major functions of WPOD-NET are summarized as follows:

i. to detect the distorted LP at non-frontal views;
ii. to estimate the distortion to unwarp the LP into a horizontally and vertically aligned object.

The OCR is implemented by OCR-NET to decode the LP number from the LP image. The OCR-NET is proven to achieve reliable performance across the LPs from different countries [Silva and Jung, 2018].

2.2 Resizing

In case of frontal/rear views, the ratio between the dimensions of LP and vehicle BB is high. However, this ratio tends to be much lower for oblique/lateral views, since the vehicle BB tends to be larger and more elongated. Hence, oblique views should be resized to a larger dimension than frontal ones to keep the LP region still recognizable. The resizing factor f_{sc} is determined from the aspect ratio of the vehicle BB as follows:

$$f_{sc} = \frac{1}{min\{W_v, H_v\}} min\left\{288 \times \frac{max(W_v, H_v)}{min\{W_v, H_v\}}, 608\right\} \quad (1)$$

where W_v and H_v denote the width and height of the vehicle BB, respectively. The scaling factors 288 and 608 are selected based on experiments to keep a good compromise between accuracy and running times [Silva and Jung, 2018].

In the ALPR pipeline, the vehicles detected by YOLOv4 are resized by f_{sc} and then fed into WPOD-NET. The LP detection performance of WPOD-NET degrades if its input image is not resized by f_{sc}. To overcome this, the WPOD-NET model should be enabled with dynamic input shape to detect LPs in the vehicle images with variable input shape.

Table 3. Software and Operating System

Prerequisite	Version
Ubuntu*	20.04.4 LTS
Docker*	20.10.16
OpenVINO™	OpenVINO™ 2022.1 and 2021.4
Python*	3.8.10
NumPy*	1.19.5
OpenCV*	4.5.5(OpenVINO™ 2022.1), 4.5.3-openvino (OpenVINO™ 2021.4)
TensorFlow*	2.5.3(OpenVINO™ 2022.1), 2.4.4(OpenVINO™ 2021.4)

§

3.0 Implementation of Unconstrained ALPR

In this section, the WPOD-NET model used for LP detection is optimized with static and dynamic input shape using OpenVINO™ 2021.4 and 2022.1, respectively. The resulting detection performance is compared between these two implementations to demonstrate the capability of dynamic input shape feature in OpenVINO™ 2022.1.

The test images used in this study are downloaded from the Cars dataset. The prerequisites including the docker command and the required dependencies for implementing the unconstrained ALPR with the two OpenVINO™ toolkit versions are covered in *Appendix A: Prerequisites for Implementing Unconstrained ALPR*.

The cropped car images fed to the WPOD-NET model are extracted by running the YOLOv4-based object detection on the input images. The source code for implementing the vehicle detection with the optimized YOLOv4 model using OpenVINO™ 2022.1 is available at https://github.com/openvinotoolkit/open_model_zoo/tree/master/demos/object_detection_demo/python.

3.1 Preparing the Model

First, download the pre-trained WPOD-NET model architecture (.json) and weights (.h5) from https://github.com/sergiomsilva/alpr-unconstrained. The model files are converted into the SavedModel format using TensorFlow as follows:

```
import tensorflow as tf
from os.path import splitext
from tensorflow.keras.models  import model_from_json

def save_model(path):
    try:
        path = splitext(path)[0]
        with open('%s.json' % path, 'r') as json_file:
            model_json = json_file.read()
        model = model_from_json(model_json,
         custom_objects={})
        model.load_weights('%s.h5' % path)
        tf.saved_model.save(model,'data/lp-detector/tf2/
         models/saved_model/')
        return model
    except Exception as e:
        print(e)

wpod_net_path = "data/lp-detector/tf2/models/wpod-net.json"
wpod_net = save_model(wpod_net_path)
```

3.2 Implementing with OpenVINO™ 2021.4

In this section, the WPOD-NET is optimized with static input shape using OpenVINO™ 2021.4 to illustrate the degradation in its detection performance on the distorted LPs.

3.2.1 Optimizing Model with Static Input Shape

As OpenVINO™ 2021.4 does not support the dynamic input shape in the optimized model, the WPOD-NET model in the SavedModel format is converted into IR files with static input shape using the Model Optimizer command-line tool.

Run the following command to create the optimized WPOD-NET model with static input shape of [1,416,416,3]:

```
python3
$INTEL_OPENVINO_DIR/deployment_tools/model_optimizer/mo_tf.
py --data_type=FP32 --saved_model_dir=./data/lp-
detector/tf2/models/saved_model/ --model_name=wpod-net --
reverse_input_channels --input_shape [1,416,416,3] --
output_dir=./data/lp-
detector/tf2/models/saved_model/FP32_416
```

Figure 3. Model Optimizer Output in OpenVINO™ 2021.4

```
root@a39b02b917d8:/home/openvino/LPR/alpr/alpr-unconstrained-master# python3 $INTEL_O
PENVINO_DIR/deployment_tools/model_optimizer/mo_tf.py --data_type=FP32 --saved_model_
dir=./data/lp-detector/tf2/models/saved_model/ --model_name=wpod-net --reverse_input_
channels --input_shape [1,416,416,3] --output_dir=./data/lp-detector/tf2/models/FP32_
416/
Model Optimizer arguments:
Common parameters:
        - Path to the Input Model:      None
        - Path for generated IR:        /home/openvino/LPR/alpr/alpr-unconstrained-ma
ster/./data/lp-detector/tf2/models/FP32_416/
        - IR output name:       wpod-net
        - Log level:    ERROR
        - Batch:        Not specified, inherited from the model
        - Input layers:         Not specified, inherited from the model
        - Output layers:        Not specified, inherited from the model
        - Input shapes:         [1,416,416,3]
        - Mean values:  Not specified
        - Scale values:         Not specified
        - Scale factor:         Not specified
        - Precision of IR:      FP32
        - Enable fusing:        True
        - Enable grouped convolutions fusing:   True
        - Move mean values to preprocess section:       None
        - Reverse input channels:       True
TensorFlow specific parameters:
        - Input model in text protobuf format:  False
        - Path to model dump for TensorBoard:   None
        - List of shared libraries with TensorFlow custom layers implementation:
       None
        - Update the configuration file with input/output node names:   None
        - Use configuration file used to generate the model with Object Detection API
:  None
        - Use the config file:  None
        - Inference Engine found in:    /opt/intel/openvino/python/python3.8/openvino
Inference Engine version:       2021.4.2-3974-e2a469a3450-releases/2021/4
Model Optimizer version:        2021.4.2-3974-e2a469a3450-releases/2021/4
[ SUCCESS ] Generated IR version 10 model.
[ SUCCESS ] XML file: /home/openvino/LPR/alpr/alpr-unconstrained-master/data/lp-detec
tor/tf2/models/FP32_416/wpod-net.xml
[ SUCCESS ] BIN file: /home/openvino/LPR/alpr/alpr-unconstrained-master/data/lp-detec
tor/tf2/models/FP32_416/wpod-net.bin
[ SUCCESS ] Total execution time: 11.19 seconds.
[ SUCCESS ] Memory consumed: 487 MB.
It's been a while, check for a new version of Intel(R) Distribution of OpenVINO(TM) t
oolkit here https://software.intel.com/content/www/us/en/develop/tools/openvino-toolk
it/download.html?cid=other&source=prod&campid=ww_2022_bu_IOTG_OpenVINO-2022-1&content
=upg_all&medium=organic or on the GitHub*
```

The resulting IR file (wpod-net.xml) shows the static input shape ([1,3,416,416]) of the optimized model in the following:

```
<?xml version="1.0" ?>
<net name="wpod-net" version="10">
   <layers>
      <layer id="0" name="input" type="Parameter"
         version="opset1">
         <data shape="1, 3, 416, 416" element_type="f32"/>
         <output>
            <port id="0" precision="FP32"
               names="Func/StatefulPartitionedCall/input/_0
               :0,input:0">
               <dim>1</dim>
               <dim>3</dim>
               <dim>416</dim>
               <dim>416</dim>
            </port>
         </output>
      </layer>
```

3.2.2 Detection Results

This section presents the source code for implementing LP detection and OCR and the detection outputs of the optimized WPOD-NET model on two test images. The source code templates for reconstructing the detected LP with WPOD-NET output and for recognizing the LP number with OCR-NET are available at https://github.com/sergiomsilva/alpr-unconstrained.

As shown in the following source code, the input image is resized to the fixed input shape (416, 416) of the optimized model. The successful LP detection and OCR outputs are shown in Figure 4(I-C) and 4(I-A) for the input image in Figure 4(I-B), where the LP is captured at nearly frontal view.

On the other hand, the LP captured at non-frontal view in Figure 4(II-B) is more distorted than Figure 4(I-B) as some of the LP characters are not clearly visible. As the input image is not resized by the resizing factor according to Eq. (1) in Section 2.2, the WPOD-NET model could not detect the LP completely (Figure 4 (II-C)) resulting in incorrect LP number (Figure 4(II-A)) with OCR-NET.

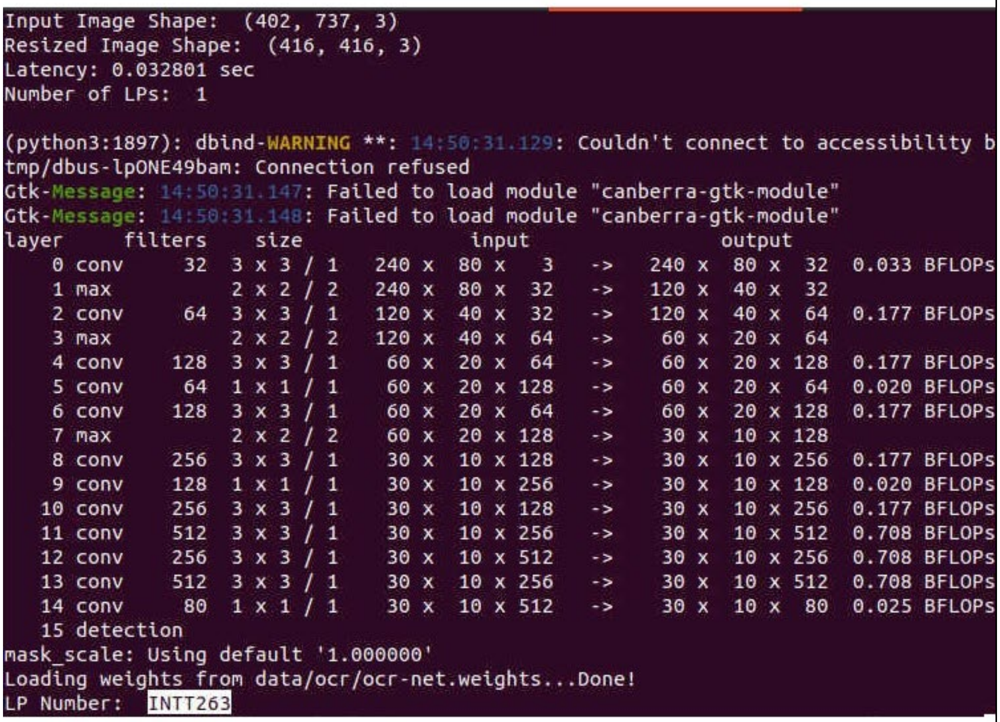
Thus, the results in Figure 4-II clearly demonstrate that the WPOD-NET model optimized with static input shape failed to detect the distorted/misaligned LP.

```
import os
import sys
import cv2
import numpy as np
from openvino.inference_engine import IECore
from src.keras_utils import reconstruct
import datetime
from lpr_ocr_net import ocr
from os.path import splitext

def ocr_net(lp_img,img_path):
    lp_gray = cv2.cvtColor(lp_img,cv2.COLOR_BGR2GRAY)
    lp_gray = (lp_gray*255).astype(np.uint8)
    cv2.imshow("License Plate",lp_gray)
    cv2.waitKey(0)
    lp_path = splitext(img_path)[0]+'_lp.png'
    cv2.imwrite(lp_path,lp_gray)
    lp_num = ocr(lp_path)
    return lp_num

def main():
    ie = IECore()
    net = ie.read_network(model="model/FP32_saved_416/wpod-
      net.xml")
    exec_net = ie.load_network(net, "CPU")
    output_layer_ir = next(iter(exec_net.outputs))
    input_layer_ir = next(iter(exec_net.input_info))
    N, C, H, W =
      net.input_info[input_layer_ir].tensor_desc.dims
    t1 = datetime.datetime.now()
    image = cv2.imread(sys.argv[1])
    print("Input Image Shape: ",image.shape)
    image = image.astype('float32')/255.
    image_resized = cv2.resize(image,(W,H))
    print("Resized Image Shape: ",image_resized.shape)
    input_tensor = np.expand_dims(image_resized,0)
    input_tensor = np.transpose(input_tensor,(0,3,1,2))
    results = exec_net.infer(inputs={input_layer_ir:
      input_tensor})
    predictions = results[output_layer_ir]
    predictions = np.squeeze(predictions)
    predictions = np.transpose(predictions, (1,2,0))
    Llp, LlpImgs = reconstruct(image, image_resized,
      predictions, (240,80), 0.5)
    t2 = datetime.datetime.now()
    latency = (t2 - t1).total_seconds()
    print("Latency: {} sec".format(latency))
    print("Number of LPs: ",len(LlpImgs))
    cv2.imshow("Input Image",image)
    cv2.waitKey(0)
    if len(LlpImgs):
        lp_txt = ocr_net(LlpImgs[0],sys.argv[1])
        print("LP Number: ",lp_txt)
    cv2.destroyAllWindows()
    return 0
```

Figure 4. Output of Unconstrained ALPR using the Optimized WPOD-NET Model with Static Input Shape

I) Case-I: Successul LP detection

A



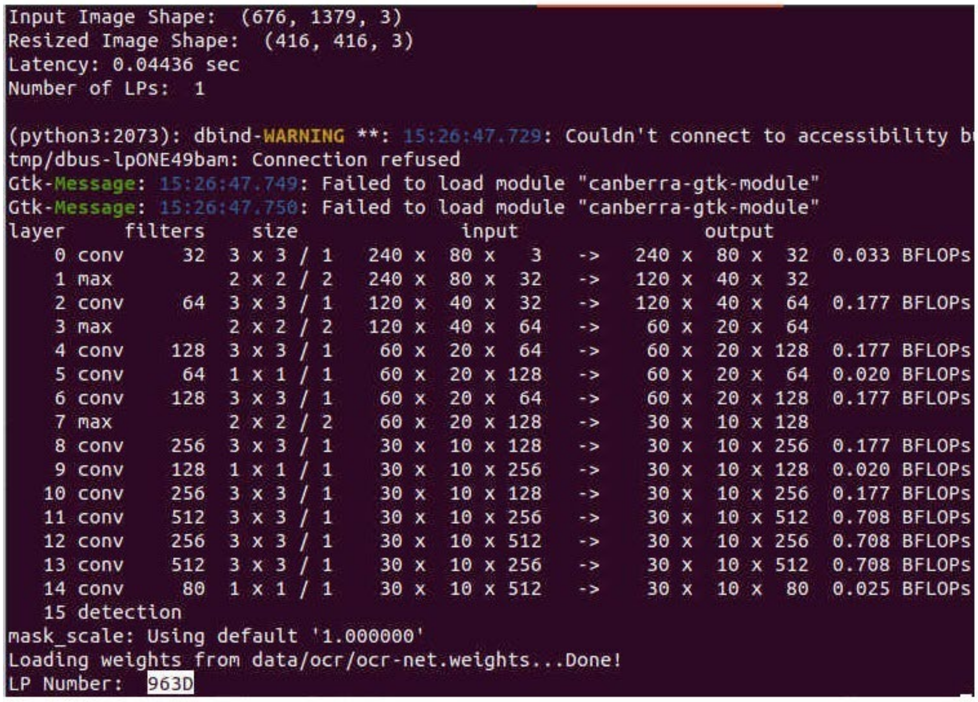
```
Input Image Shape:  (402, 737, 3)
Resized Image Shape:  (416, 416, 3)
Latency: 0.032801 sec
Number of LPs:  1

(python3:1897): dbind-WARNING **: 14:50:31.129: Couldn't connect to accessibility b
tmp/dbus-lpONE49bam: Connection refused
Gtk-Message: 14:50:31.147: Failed to load module "canberra-gtk-module"
Gtk-Message: 14:50:31.148: Failed to load module "canberra-gtk-module"
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   240 x  80 x   3   ->   240 x  80 x  32  0.033 BFLOPs
    1 max          2 x 2 / 2   240 x  80 x  32   ->   120 x  40 x  32
    2 conv     64  3 x 3 / 1   120 x  40 x  32   ->   120 x  40 x  64  0.177 BFLOPs
    3 max          2 x 2 / 2   120 x  40 x  64   ->    60 x  20 x  64
    4 conv    128  3 x 3 / 1    60 x  20 x  64   ->    60 x  20 x 128  0.177 BFLOPs
    5 conv     64  1 x 1 / 1    60 x  20 x 128   ->    60 x  20 x  64  0.020 BFLOPs
    6 conv    128  3 x 3 / 1    60 x  20 x  64   ->    60 x  20 x 128  0.177 BFLOPs
    7 max          2 x 2 / 2    60 x  20 x 128   ->    30 x  10 x 128
    8 conv    256  3 x 3 / 1    30 x  10 x 128   ->    30 x  10 x 256  0.177 BFLOPs
    9 conv    128  1 x 1 / 1    30 x  10 x 256   ->    30 x  10 x 128  0.020 BFLOPs
   10 conv    256  3 x 3 / 1    30 x  10 x 128   ->    30 x  10 x 256  0.177 BFLOPs
   11 conv    512  3 x 3 / 1    30 x  10 x 256   ->    30 x  10 x 512  0.708 BFLOPs
   12 conv    256  3 x 3 / 1    30 x  10 x 512   ->    30 x  10 x 256  0.708 BFLOPs
   13 conv    512  3 x 3 / 1    30 x  10 x 256   ->    30 x  10 x 512  0.708 BFLOPs
   14 conv     80  1 x 1 / 1    30 x  10 x 512   ->    30 x  10 x  80  0.025 BFLOPs
   15 detection
mask_scale: Using default '1.000000'
Loading weights from data/ocr/ocr-net.weights...Done!
LP Number:  INTT263
```

B

C

II) Case-II: Failed LP detection

A



```
Input Image Shape:  (676, 1379, 3)
Resized Image Shape:  (416, 416, 3)
Latency: 0.04436 sec
Number of LPs:  1

(python3:2073): dbind-WARNING **: 15:26:47.729: Couldn't connect to accessibility b
tmp/dbus-lpONE49bam: Connection refused
Gtk-Message: 15:26:47.749: Failed to load module "canberra-gtk-module"
Gtk-Message: 15:26:47.750: Failed to load module "canberra-gtk-module"
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   240 x  80 x   3   ->   240 x  80 x  32  0.033 BFLOPs
    1 max          2 x 2 / 2   240 x  80 x  32   ->   120 x  40 x  32
    2 conv     64  3 x 3 / 1   120 x  40 x  32   ->   120 x  40 x  64  0.177 BFLOPs
    3 max          2 x 2 / 2   120 x  40 x  64   ->    60 x  20 x  64
    4 conv    128  3 x 3 / 1    60 x  20 x  64   ->    60 x  20 x 128  0.177 BFLOPs
    5 conv     64  1 x 1 / 1    60 x  20 x 128   ->    60 x  20 x  64  0.020 BFLOPs
    6 conv    128  3 x 3 / 1    60 x  20 x  64   ->    60 x  20 x 128  0.177 BFLOPs
    7 max          2 x 2 / 2    60 x  20 x 128   ->    30 x  10 x 128
    8 conv    256  3 x 3 / 1    30 x  10 x 128   ->    30 x  10 x 256  0.177 BFLOPs
    9 conv    128  1 x 1 / 1    30 x  10 x 256   ->    30 x  10 x 128  0.020 BFLOPs
   10 conv    256  3 x 3 / 1    30 x  10 x 128   ->    30 x  10 x 256  0.177 BFLOPs
   11 conv    512  3 x 3 / 1    30 x  10 x 256   ->    30 x  10 x 512  0.708 BFLOPs
   12 conv    256  3 x 3 / 1    30 x  10 x 512   ->    30 x  10 x 256  0.708 BFLOPs
   13 conv    512  3 x 3 / 1    30 x  10 x 256   ->    30 x  10 x 512  0.708 BFLOPs
   14 conv     80  1 x 1 / 1    30 x  10 x 512   ->    30 x  10 x  80  0.025 BFLOPs
   15 detection
mask_scale: Using default '1.000000'
Loading weights from data/ocr/ocr-net.weights...Done!
LP Number:  963D
```

B

C

3.3 Implementing with OpenVINO™ 2022.1

In this section, the WPOD-NET is optimized with dynamic input shape using OpenVINO™ 2022.1 to demonstrate the improved performance of LP detection and OCR models in unconstrained ALPR.

3.3.1 Optimizing Model with Dynamic Input Shape

With OpenVINO™ 2022.1, the WPOD-NET model is converted from the SavedModel format into IR files with dynamic input shape using the Model Optimizer command-line tool as shown in Figure 5.

Figure 5. Model Optimizer Output in OpenVINO™ 2022.1

```
root@99a1f3c750c5:/home/openvino/LPR/alpr/alpr-unconstrained-master# mo --saved_model_dir ./data/lp-d
etector/tf2/models/saved_model/ --output_dir ./data/lp-detector/tf2/models/FP32/
Model Optimizer arguments:
Common parameters:
        - Path to the Input Model:      None
        - Path for generated IR:        /home/openvino/LPR/alpr/alpr-unconstrained-master/./data/lp-d
etector/tf2/models/FP32/
        - IR output name:       saved_model
        - Log level:    ERROR
        - Batch:        Not specified, inherited from the model
        - Input layers:         Not specified, inherited from the model
        - Output layers:        Not specified, inherited from the model
        - Input shapes:         Not specified, inherited from the model
        - Source layout:        Not specified
        - Target layout:        Not specified
        - Layout:       Not specified
        - Mean values:  Not specified
        - Scale values:         Not specified
        - Scale factor:         Not specified
        - Precision of IR:      FP32
        - Enable fusing:        True
        - User transformations:         Not specified
        - Reverse input channels:       False
        - Enable IR generation for fixed input shape:   False
        - Use the transformations config file:  None
Advanced parameters:
        - Force the usage of legacy Frontend of Model Optimizer for model conversion into IR:   False
        - Force the usage of new Frontend of Model Optimizer for model conversion into IR:      False
TensorFlow specific parameters:
        - Input model in text protobuf format:  False
        - Path to model dump for TensorBoard:   None
        - List of shared libraries with TensorFlow custom layers implementation:        None
        - Update the configuration file with input/output node names:   None
        - Use configuration file used to generate the model with Object Detection API:  None
        - Use the config file:  None
OpenVINO runtime found in:      /opt/intel/openvino/python/python3.8/openvino
OpenVINO runtime version:       2022.1.0-7019-cdb9bec7210-releases/2022/1
Model Optimizer version:        2022.1.0-7019-cdb9bec7210-releases/2022/1
[ WARNING ]  The model contains input(s) with partially defined shapes: name="input" shape="[-1, -1,
-1, 3]". Starting from the 2022.1 release the Model Optimizer can generate an IR with partially defin
ed input shapes ("-1" dimension in the TensorFlow model or dimension with string value in the ONNX mo
del). Some of the OpenVINO plugins require model input shapes to be static, so you should call "resha
pe" method in the Inference Engine and specify static input shapes. For optimal performance, it is st
ill recommended to update input shapes with fixed ones using "--input" or "--input_shape" command-lin
e parameters.
[ SUCCESS ] Generated IR version 11 model.
[ SUCCESS ] XML file: /home/openvino/LPR/alpr/alpr-unconstrained-master/data/lp-detector/tf2/models/F
P32/saved_model.xml
[ SUCCESS ] BIN file: /home/openvino/LPR/alpr/alpr-unconstrained-master/data/lp-detector/tf2/models/F
P32/saved_model.bin
[ SUCCESS ] Total execution time: 10.48 seconds.
[ SUCCESS ] Memory consumed: 518 MB.
It's been a while, check for a new version of Intel(R) Distribution of OpenVINO(TM) toolkit here http
s://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit/download.html?cid=other&sourc
e=prod&campid=ww_2022_bu_IOTG_OpenVINO-2022-1&content=upg_all&medium=organic or on the GitHub*
[ INFO ] The model was converted to IR v11, the latest model format that corresponds to the source DL
 framework input/output format. While IR v11 is backwards compatible with OpenVINO Inference Engine A
PI v1.0, please use API v2.0 (as of 2022.1) to take advantage of the latest improvements in IR v11.
Find more information about API v2.0 and IR v11 at https://docs.openvino.ai
```

Run the following command to create the optimized WPOD-NET model:

```
mo --saved_model_dir ./data/lp-
detector/tf2/models/saved_model/ --output_dir ./data/lp-
detector/tf2/models/FP32/
```

The resulting IR file (saved_model.xml) shows the dynamic input shape ([?, ?, ?, 3]) of the optimized model in the following:

```
<?xml version="1.0" ?>
<net name="saved_model" version="11">
   <layers>
      <layer id="0" name="input" type="Parameter"
         version="opset1">
         <data shape="?,?,?,3" element_type="f32"/>
         <rt_info>
            <attribute name="fused_names" version="0"
               value="input"/>
            <attribute name="old_api_map_order" version="0"
               value="0, 2, 3, 1"/>
         </rt_info>
         <output>
            <port id="0" precision="FP32"
               names="Func/StatefulPartitionedCall/input/_0
               :0,input:0">
               <dim>-1</dim>
               <dim>-1</dim>
               <dim>-1</dim>
               <dim>3</dim>
            </port>
         </output>
      </layer>
```

The input shape ([?, ?, ?, 3]) in the above IR file indicates that the dynamic shape feature in OpenVINO™ 2022.1 is able to preserve the undefined input dimensions of the raw WPOD-NET model ([null, null, null, 3]). This enables the optimized model to dynamically change its input shape according to the input images with variable input shape.

3.3.2 Detection Results

This section presents the source code and the detection output of the optimized WPOD-NET model with dynamic input shape on the distorted LP image using OpenVINO™ 2022.1.

As shown in the preprocess function in the following source code, the input image is resized by the resizing factor according to Eq. (1) in Section 2.2. The main function implements the LP detection on the resized image with the optimized WPOD-NET model. From the console output in Figure 6A, it is evident that the input image in Figure 6B is resized from [676, 1379, 3] to [608, 1232, 3]. As a result, the distorted LP is completely detected and transformed into a horizontally aligned object (Figure 6C) resulting in the correct LP number with OCR-NET (Figure 6A).

```
import os
import sys
import cv2
import numpy as np
from openvino.runtime import Core, PartialShape
from src.keras_utils import reconstruct
import datetime
from lpr_ocr_net import ocr
from os.path import splitext

def preprocess(img):
    print("Input Image Shape: ",img.shape)
    max_dim = max(img.shape[:2])
    min_dim = min(img.shape[:2])
    net_stride = 2**4
    ratio = float(max_dim)/min_dim
    side = int(ratio*288.)
    bound_dim = min(side + (side % (net_stride)), 608)
    factor = float(bound_dim)/min_dim
    w,h = (np.array(img.shape[1::-
      1],dtype=float)*factor).astype(int).tolist()
    w += (w%net_stride!=0)*(net_stride - w%net_stride)
    h += (h%net_stride!=0)*(net_stride - h%net_stride)
    img_resized = cv2.resize(img, (w, h))
    print("Resized Image Shape: ",img_resized.shape)
    return img_resized

def main():
    ie = Core()
    model = ie.read_model("data/lp-
      detector/tf2/models/FP32/saved_model.xml")
    compiled_model = ie.compile_model(model=model,
      device_name="CPU")
    t1 = datetime.datetime.now()
    image = cv2.imread(sys.argv[1])
    image = image.astype('float32') / 255.0
    image_resized = preprocess(image)
    input_tensor = np.expand_dims(image_resized, 0)
    results = compiled_model.infer_new_request({0:
      input_tensor})
    predictions = next(iter(results.values()))
    predictions = np.squeeze(predictions)
    Llp, LlpImgs = reconstruct(image, image_resized,
      predictions, (240,80), 0.5)
    t2 = datetime.datetime.now()
    latency = (t2 - t1).total_seconds()
    print("Latency: {} sec".format(latency))
    print("Number of LPs: ",len(LlpImgs))
    cv2.imshow("Input Image",image)
    cv2.waitKey(0)
    if len(LlpImgs):
        lp_txt = ocr_net(LlpImgs[0],sys.argv[1])
        print("LP Number: ",lp_txt)
    cv2.destroyAllWindows()
    return 0
```

Figure 6. Output of Unconstrained ALPR using the Optimized WPOD-NET Model with Dynamic Input Shape

A

```
Input Image Shape:  (676, 1379, 3)
Resized Image Shape:  (608, 1232, 3)
Latency: 0.155021 sec
Number of LPs:  1
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   240 x  80 x   3   ->   240 x  80 x  32  0.033 BFLOPs
    1 max          2 x 2 / 2   240 x  80 x  32   ->   120 x  40 x  32
    2 conv     64  3 x 3 / 1   120 x  40 x  32   ->   120 x  40 x  64  0.177 BFLOPs
    3 max          2 x 2 / 2   120 x  40 x  64   ->    60 x  20 x  64
    4 conv    128  3 x 3 / 1    60 x  20 x  64   ->    60 x  20 x 128  0.177 BFLOPs
    5 conv     64  1 x 1 / 1    60 x  20 x 128   ->    60 x  20 x  64  0.020 BFLOPs
    6 conv    128  3 x 3 / 1    60 x  20 x  64   ->    60 x  20 x 128  0.177 BFLOPs
    7 max          2 x 2 / 2    60 x  20 x 128   ->    30 x  10 x 128
    8 conv    256  3 x 3 / 1    30 x  10 x 128   ->    30 x  10 x 256  0.177 BFLOPs
    9 conv    128  1 x 1 / 1    30 x  10 x 256   ->    30 x  10 x 128  0.020 BFLOPs
   10 conv    256  3 x 3 / 1    30 x  10 x 128   ->    30 x  10 x 256  0.177 BFLOPs
   11 conv    512  3 x 3 / 1    30 x  10 x 256   ->    30 x  10 x 512  0.708 BFLOPs
   12 conv    256  3 x 3 / 1    30 x  10 x 512   ->    30 x  10 x 256  0.708 BFLOPs
   13 conv    512  3 x 3 / 1    30 x  10 x 256   ->    30 x  10 x 512  0.708 BFLOPs
   14 conv     80  1 x 1 / 1    30 x  10 x 512   ->    30 x  10 x  80  0.025 BFLOPs
   15 detection
mask_scale: Using default '1.000000'
Loading weights from data/ocr/ocr-net.weights...Done!
LP Number:  J963FPS
```

Table 4. Comparison of Detection Latency between Raw and Optimized WPOD-NET Models

Input Shape	Detection Latency (s)		
	Raw Model	OpenVINO™ 2021.4	OpenVINO™ 2022.1
Static	0.56	0.045	0.044
Dynamic	0.84	-	0.17

The detection latency in Table 4 includes the time taken for preprocessing the input image and detecting the LP with WPOD-NET model.

For the model with static input shape (416x416), the detection latency is decreased by 92% with the optimized model (using OpenVINO™ 2021.4 or 2022.1) compared to the raw model (0.56 s).

On the other hand, the model optimized with dynamic input shape using OpenVINO™ 2022.1 is able to detect the distorted LP, while decreasing the detection latency by 80% compared to the raw model (0.84 s).

§

4.0 Conclusion

The dynamic input shape feature introduced in OpenVINO™ 2022.1 enables the optimized model to process the input images whose shape changes before executing the inference at runtime.

To demonstrate its capability, this white paper presented the optimized implementation of unconstrained ALPR that detects and aligns the distorted LPs using WPOD-NET model. The unconstrained ALPR is applicable to use cases such as monitoring the traffic violations and tracking the vehicles in congested roads and residential areas, where the vehicles are often captured at non-frontal views due to oblique shooting angles.

The method of optimizing WPOD-NET with dynamic input shape using OpenVINO™ 2022.1 is presented to show that the undefined input dimensions of the raw model are preserved in the optimized model. This enables the optimized model to reshape its input according to the input image being resized at runtime. As a result, the optimized model with dynamic input shape achieves better detection performance on the distorted LPs compared to the model optimized with static input shape using OpenVINO™ 2021.4. This also improves the overall performance of unconstrained ALPR as the succeeding character recognition performs well only on the aligned LP images. Furthermore, the model optimized with dynamic input shape decreased the detection latency by 80% compared to the raw model.

§

5.0 Appendix A: Prerequisites for Implementing Unconstrained ALPR

1. Download the pre-trained WPOD-NET and OCR-NET models and the source code template for implementing the reconstruct and ocr functions from https://github.com/sergiomsilva/alpr-unconstrained. Save them in the directory to be mapped to the container.

2. Pull the docker images and create the respective container.

```
docker pull openvino/ubuntu20_data_dev:2021.4.2
docker pull openvino/ubuntu20_dev:2022.1.0

docker run -u=0 -it --name openvino2021_4_ubuntu20 --device
/dev/dri:/dev/dri --device-cgroup-rule='c 189:* rmw' -v
/dev/bus/usb:/dev/bus/usb -v
~/.Xauthority:/root/.Xauthority -v /tmp/.X11-
unix/:/tmp/.X11-unix/ -e DISPLAY=$DISPLAY -v
/home/ramesh/Documents/openvino2021_4_ubuntu20:/home/openvi
no openvino/ubuntu20_data_dev:2021.4.2

docker run -u=0 -it --name openvino2022_1_ubuntu20 --device
/dev/dri:/dev/dri --device-cgroup-rule='c 189:* rmw' -v
/dev/bus/usb:/dev/bus/usb -v
~/.Xauthority:/root/.Xauthority -v /tmp/.X11-
unix/:/tmp/.X11-unix/ -e DISPLAY=$DISPLAY -v
/home/ramesh/Documents/openvino2022_1_ubuntu20:/home/openvi
no openvino/ubuntu20_dev:2022.1.0
```

3. Install the required dependencies in the container.

```
apt update
apt install sudo
sudo apt install vim qt5-default
```

§