License Plate Recognition using Training Toolbox based on LPRNet Model

Implementation Guide

March 2019

Contents

List of Abbreviations

Abbreviation	Expanded Form
API	Application Program Interface
CNN	Convolutional Neural Network
LPRNet	License Plate Recognition Network
FPGA	Field Programmable Gate Array
IR	Intermediate Representation
Open VINO	Open Visual Inferencing and Neural Network Optimization
OpenCL	Open Computing Language
OpenCV	Open Computer Vision
POC	Proof of Concept
LP	License Plate

1. Introduction

There is a need to design, develop, and test license plates recognition and vehicle attributes detection prototype to demonstrate the feasibility of Intel® Distribution of OpenVINO™ toolkit and LPRNet TensorFlow* training toolbox. The Intel Distribution of OpenVINO toolkit pre-trained model scales to detect vehicle and corresponding vehicle attributes such as color, license plate recognition and detection with sufficient accuracy and performance for a trained scenario. Keeping in mind of variation in the scenarios, these pre-trained models are open-sourced so that you can use it to train on your own dataset and use it efficiently in your environment.

The toolkit is a comprehensive toolkit for quickly developing applications and solutions that emulate human vision. Based on CNNs, the toolkit extends computer vision workloads across Intel® hardware, maximizing performance.

The toolkit:

- Enables the CNN-based deep learning inference on the edge.
- Supports heterogeneous execution across Intel's computer vision accelerators, using a common API for the CPU, Integrated Graphics, Intel® Movidius™ Neural Compute Stick (NCS), and FPGA. However, you are not covering Intel Movidius NCS and FPGA in this lab.
- Speeds time-to-market through an easy-to-use library of computer vision functions and pre-optimized kernels.
- Includes optimized calls for computer vision standards, including OpenCV, OpenCL™, and OpenVX*.

2. System Requirements

Hardware Requirements

- Processor : x86_64 is the only supported architecture
- Memory : At least 2GB RAM
- Hard Disk : 32GB HDD
- Network : Network adapter with active internet connection

Software Requirements

- Operating System : Ubuntu* 16.04 LTS 64 bit
- Git
- Intel Distribution of OpenVINO toolkit
- Python* 3.4-3.6
- TensorFlow

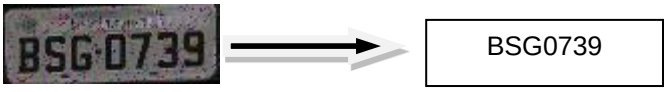
3. About LPRNet

Automatic License Plate Recognition is a challenging and important task which is used in traffic management, digital security surveillance, vehicle recognition, parking management of big cities. This task is a complex problem due to many factors which include but are not limited to: blurry images, poor lighting conditions, variability of license plates numbers (including special characters logograms for China, Japan), physical impact (deformations), weather conditions.

LPRNet is based on Deep Convolutional Neural Network. Recent studies proved effectiveness and superiority of Convolutional Neural Networks in many Computer Vision tasks such as image classification, object detection and semantic segmentation. However, running most of them on embedded devices still remains a challenging problem.

Let us summarize the LPRNet as follows:

- LPRNet is a real-time framework for high-quality license plate recognition supporting template and character independent variable-length license plates, performing LPR without character pre-segmentation, trainable end-to-end from scratch for different national license plates.
- LPRNet is the first real-time approach that does not use Recurrent Neural Networks and is light in weight enough to run on variety of platforms, including embedded devices.
- Application of LPRNet to real traffic surveillance video shows that our approach is robust enough to handle difficult cases, such as perspective and camera-dependent distortions, hard lighting conditions, change of viewpoint, and so on.



4. Retraining LPRNet

This section describes how to retrain LPRNet model with custom datasets of any country license plate with reference to LPRNet training toolbox for Chinese license plate. For example, taking synthetic Brazil license plates. The following steps are involved for retraining:

- Downloading the LPRNet repository
- Preprocessing the datasets
- Package adjustments
- Editing the configuration file
- Training the model
- Evaluating the trained model

After completing these steps, you can retrain LPRNet model and recognize license plates of any nation.

Note In this process, you are not using virtual environment setup as described in the git repositories of the training toolbox.

4.1. Downloading the LPRNet Repository

Download ***openvino_training_extensions*** repository under develop branch which contains LPRNet and SSD_detector for the Chinese Car using GitHub* link. Setup the toolkit training extensions using the quick start guide given in the link.

When you visit the LPRnet specific folder in the link in training toolbox, you will see the steps to train the LPRNet model for Chinese LP.

Note:

1. Don’t download the master repository because there are some files missing so download the repository under the develop branch.
2. There could be some change in the folder structures then mentioned in the documents as training toolbox project get updated by owner time to time.

Extract the downloaded repository to desired directory. Next you need to download the datasets (images) for retraining the model by default the LPRNet model provides a link to download the Chinese license plates which has around 300000 images. So download the datasets which you want to retrain the model with.

Copy the downloaded datasets to the ***<LPRNet extracted directory>/openvino_training_extensions-develop/data/synthetic_chinese_license_plates/crops/*** directory

4.2. Preprocessing the Datasets

Next step is to perform the dataset preprocessing which includes the following:

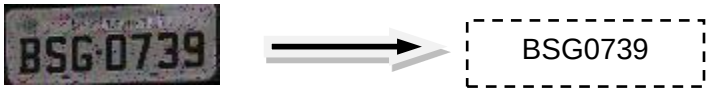
- Annotating the datasets
- Split the datasets into train and validation labels

4.2.1. Annotating the Datasets

Navigate to ***openvino_training_extensions-develop/data/synthetic_chinese_license_plates/*** and write the annotations for the images in **annotation** text file. The annotation file should have all the image paths followed by the number plate data. The license plate data can have special symbols followed by alphabets/numbers.

For example: *<path_to_the_image> <special_symbol><remaing_Alphabets/numbers>*

E.g. *home/intel/Downloads/Datasets/License_plates/000001.jpg BSG0739*



000001.jpg

4.2.2. Split the Datasets into Train and Validation Labels

Access the following ***openvino_training_extensions-develop/data/synthetic_chinese_license_plates/*** directory and execute the script ***make_train_split.py*** to split into train and Val files. These are used for training and validation. This script automatically splits the entire datasets in the ratio 9:1 for training and validation.

Execute the python file by using the following command:

```
$ python3 make_train_split.py home/intel/openvino_training_extensions-develop/data/synthetic_chinese_license_plates/annotation
```

You can view two files named ***train*** and ***val*** which are annotation forms of train and validation files.

4.3. Package Adjustments

As the next step, you need to make some adjustments or workarounds because there are some missing files in the downloaded repository. You need to copy them in their respective places since the repository is in the development stage. It is not 100 percent deployable. Perform the following changes:

- Script Placement Adjustments
- Package Adjustments(workaround)
- Workaround Package Initiation

4.3.1. Script Placement Adjustments

Navigate to the ***openvino_training_extensions-develop/tensorflow_toolkit/lpr*** directory and move the train.py, eval.py, export.py, infer.py, infer_ie.py to the ***openvino_training_extensions-develop/tensorflow_toolkit*** directory. You need to perform this because the individual scripts are looking for packages in the moved directory.

4.3.2. Package Adjustments (Workaround)

Copy the script ***spatial_transformer.py*** from the GitHub page. Paste it in ***openvino_training_extensions-develop/tensorflow_toolkit/lpr/toolbox/*** directory. At the time of release of this repository, there was no spatial_transformer.py script in the corresponding folder. So, you need to explicitly download the file and place it in the corresponding directory.

4.3.3. Workaround Package Initiation

Go to ***openvino_training_extensions-develop/tensorflow_toolkit/lpr*** directory, and open **trainer.py** file and change the line 7 command ***spatial_transformer path*** *to* ***lpr.toolbox.spatial_transformer***. Because the train script is checking for the spatial_transformer file so in the previous step you downloaded the file and initiated the script.

4.4. Editing the Configuration file

You need to make some changes in the configuration file as the next step. This is to point to the correct folders for training our datasets.

Editing is done for the following files:

- Editing the config file
- Editing the main logic for training our custom datasets

4.4.1. Editing the config file

Go to ***openvino_training_extensions-develop/tensorflow_toolkit/lpr/chinese_lp*** directory and open ***config.py*** file and change the train path to ***<root_directory>/openvino_training_extensions-develop/data/synthetic_chinese_license_plates/train*** under ***train class***.

This is to make the script aware of where the train annotation file is located during training our datasets.

4.4.2. Editing the main logic for training our custom datasets

Open **openvino_training_extensions-develop/tensorflow_toolkit/lpr/toolbox/utils.py, look for the definition of the** variable 'lpr_patterns' and set the corresponding regular expression based on the annotation labels you used for datasets.

Ex: '^ [A-Z]{3}[0-9]{4}$' for labels starting with 3 alphabets and followed by four numericals.

If there are special symbols used in datasets then you can refer the Chinese license plate regular expressions which are by default given. If you are not using any special symbols then comment out those lines

4.5. Training the Model

The next step is very important since you made all the configurations and did all the workarounds (folder structure) in the previous step. Go to **openvino_training_extensions-develop/tensorflow_toolkit/** directory and execute the command for training.

```
$ python3 train.py lpr/chinese_lp/config.py
```

After training is over you can see the checkpoints in the **model_test** directory.

4.6. Evaluating the Trained Model

Go to **openvino_training_extensions-develop/tensorflow_toolkit/lpr/chinese_lp** directory and open **config.py** file and change the '**val path**' to **openvino_training_extensions-develop/data/synthetic_chinese_license_plates/val**. Under **eval** class and change the **checkpoint path** to **openvino_training_extensions-develop/tensorflow_toolkit/lpr/chinese_lp/model_test/model.ckpt-68500.ckpt** i.e., the latest checkpoint file generated while training the model in the previous step.

Go to ***/home/intel/openvino_training_extensions-develop/tensorflow_toolkit/*** directory and execute the following command to evaluate.

```
$ python3 eval.py lpr/chinese_lp/config.py
```

After evaluating, you may get the **test accuracy above 90%.** It may differ depending on the number of datasets, steps and so on.

5. Inferencing the Trained Model

Inferencing the trained model which you got from the previous steps can be done in two ways:

- Inferencing using LPRNet inference script
- Inferencing using toolkit Inference engine

5.1. Inferencing using LPRNet Inference Script

Make the similar changes in the config file under the infer class i.e., giving the checkpoint directory and path to the text file. You have to create a text file manually which has the complete path to the images you want to test or infer separated by newline.

E.g. /home/intel/Downloads/Datasets/test/01.jpg

/home/intel/Downloads/Datasets/test/02.jpg

……

……..

Go to ***/home/intel/openvino_training_extensions-develop/tensorflow_toolkit/*** directory and execute the following command to check the license plate recognition.

```
$ python3 infer.py lpr/chinese_lp/config.py
```

An output screen with the license plate image and the corresponding recognized license plate number appears.

5.2. Inferencing using Intel® Distribution of OpenVINO™ toolkit Inference Engine

In this LPRNet, you can do the inferencing using the toolkit Inference engine as a backend. Perform the following steps, to do this:

Go to ***/home/intel/openvino_training_extensions-develop/tensorflow_toolkit/*** and execute the script **export.py** by providing the required arguments.

```
$ python3 export.py lpr/chinese_lp/config.py
/opt/intel/computer_vision_sdk_2018.5.431/deployment_tools/model_optimizer/mo.py
```

We can find the corresponding .xml, .bin and frozen.pb file are being generated in ***model_test/ie_model*** directory.

These .xml and .bin files are given as input to the ‘infer_ie.py’ in ***/home/intel/openvino_training_extensions-develop/tensorflow_toolkit/*** directory to get the output. Execute the following to perform this activity.

```
$ python3 infer_ie.py --
model=lpr/chinese_lp/model_test/ie_model/graph.xml
lpr/chinese_lp/config.py
../data/synthetic_chinese_license_plates/AYU4716.jpg --
cpu_extension=/opt/intel/computer_vision_sdk_2018.5.431/deployment_too
ls/inference_engine/samples/build/intel64/Release/lib/libcpu_extension
.so
```

An output screen with the license plate image and the corresponding recognized license plate number appears.

5.3. Hardware that support the LPRNet Model using the Intel® Distribution of OpenVINO™ toolkit

- Intel® CPU x86_64
- Integrated Graphics from Intel
- FPGA
- Intel® Movidius™ Vision Processing Unit (VPU)

6. Reference Section

1. Intel Distribution of OpenVINO toolkit: https://docs.openvinotoolkit.org/latest/_docs_install_guides_installing_openvino_linux.html
2. LPRNet: https://arxiv.org/pdf/1806.10447.pdf
3. LPRNet training toolbox and SSD_detector for Chinese license plate: https://github.com/opencv/openvino_training_extensions/tree/develop/tensorflow_toolkit/lpr
4. spatial_transformer.py: https://github.com/opencv/openvino_training_extensions/blob/91c1bd4566dc152c019d12d79abd200f642c2efa/training_toolbox/lpr/toolbox/spatial_transformer_layer.py