Custom YOLO Model in the DeepStream YOLO App

Application Note

DA_09591-001 | August 2020

Document History

Doc_Number

Version	Date	Authors	Description of Change
1.0	August 6, 2019	cshah, jachs	Initial release
2.0	August 3, 2020	Chandrahassj , Sraghunath	5.0 GA Release

How to Use the Custom YOLO Model

The `objectDetector_Yolo` sample application provides a working example of the open source YOLO models: YOLOv2, YOLOv3, tiny YOLOv2, and tiny YOLOv3, and YOLOV3-SPP. You can find more information about the models at https://pjreddie.com/darknet/yolo/. The sample also illustrates NVIDIA® TensorRT™ INT8 calibration (`yolov3-calibration.table.trt7.0`).

To set up the sample

Compile the open source model and run the DeepStream app as explained by the README in `objectDetector_Yolo`. This is a sanity check that you are able to run the open source YOLO model with the sample app.

To use the custom YOLOv3 and tiny YOLOv3 models

1. Open `nvdsinfer_custom_impl_Yolo/nvdsparsebbox_Yolo.cpp`.
2. Change the value of the `NUM_CLASSES_YOLO` constant to reflect the number of classes in your model. For example, if your model uses 80 classes:

   ```
   static const int NUM_CLASSES_YOLO = 80;
   ```

   The default values in the file are from:

   https://pjreddie.com/media/files/papers/YOLOv3.pdf
   https://raw.githubusercontent.com/pjreddie/darknet/master/cfg/yolov3.cfg
   https://raw.githubusercontent.com/pjreddie/darknet/master/cfg/yolov3-tiny.cfg
   https://raw.githubusercontent.com/pjreddie/darknet/master/cfg/yolov3-spp.cfg

3. Replace the model parameters with your new model parameters in `NvDsInferParseCustomYoloV3()` (if you are using the YOLOv3) or `NvDsInferParseCustomYoloV3Tiny()` (if you are using tiny YOLOv3). Taking YOLOv3 as an example:

   ```
   extern "C" bool NvDsInferParseCustomYoloV3(
           std::vector<NvDsInferLayerInfo> const& outputLayersInfo,
           NvDsInferNetworkInfo const& networkInfo,
           NvDsInferParseDetectionParams const& detectionParams,
           std::vector<NvDsInferParseObjectInfo>& objectList)
   {
     . . .
   ```

```
    ## 9 clusters from COCO dataset
    const std::vector<float> kANCHORS =
            {10.0, 13.0, 16.0, 30.0, 33.0, 23.0, 30.0, 61.0, 62.0,
            45.0, 59.0, 119.0, 116.0, 90.0, 156.0, 198.0, 373.0, 326.0};

    ## Specifies which of the 9 anchors above to use
    static const std::vector<std::vector<int>> kMASKS = {
            {6, 7, 8},
            {3, 4, 5},
            {0, 1, 2}};
}
```

4. Update the corresponding NMS IOU Threshold and confidence threshold in the nvinfer plugin config file. Make sure to set "cluster-mode=2" to select NMS algorithm.

```
[class-attrs-all]
nms-iou-threshold=0.3
pre-cluster-threshold=0.7
```

To use custom models of YOLOv2 and YOLOv2-tiny

1. Open `nvdsinfer_custom_impl_Yolo/nvdsparsebbox_Yolo.cpp`.
2. Change the value of the `NUM_CLASSES_YOLO` constant to reflect the number of classes in your model. For example, if your model uses 80 classes:

```
static const int NUM_CLASSES_YOLO = 80;
```

The default values in the file are from:

https://raw.githubusercontent.com/pjreddie/darknet/master/cfg/yolov2.cfg
https://raw.githubusercontent.com/pjreddie/darknet/master/cfg/yolov2-tiny.cfg

3. Change the model parameters for `NvDsInferParseCustomYoloV2()` (if you are using YOLOv2) or `NvDsInferParseCustomYoloV2Tiny()` (if you are using tiny YOLOv2). Taking YOLOv2 as an example:

```
# specify NMS and confidence threshold
static const float kNMS_THRESH = 0.3f;
static const float kPROB_THRESH = 0.6f;

# specify anchors and in NvDsInferParseYoloV2, kANCHORS = {[anchors] in
yolov2.cfg} * stride
static const std::vector<float> kANCHORS = {
        18.3273602, 21.6763191, 59.9827194, 66.0009613,
        106.829758, 175.178879, 252.250244, 112.888962,
        312.656647, 293.384949 };
# Predicted boxes in NvDsInferParseYoloV2
const uint kNUM_BBOXES = 5;
```

4. Update the corresponding NMS IOU Threshold and confidence threshold in the nvinfer plugin config file. Make sure to set “cluster-mode=2” to select NMS algorithm.

```
[class-attrs-all]
nms-iou-threshold=0.3
pre-cluster-threshold=0.6
```

Note: The built-in example ships with the TensorRT INT8 calibration file `yolov3-calibration.table.trt7.0`. The example runs at INT8 precision for best performance. To compare the performance to the built-in example, generate a new INT8 calibration file for your model.

You can run the sample with another type of precision, but it will be slower. If you run with FP16 or FP32 precision, change the `network-mode` parameter in the configuration file `(config_infer_primary_yolo*.txt`.

```
## 0=FP32, 1=INT8, 2=FP16 mode
network-mode=1 <== Change to 0 or 2
```

Notice

The information provided in this specification is believed to be accurate and reliable as of the date provided. However, NVIDIA Corporation (“NVIDIA”) does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This publication supersedes and replaces all other specifications for the product that may have been previously supplied.
NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and other changes to this specification, at any time and/or to discontinue any product or service without notice. Customer should obtain the latest relevant specification before placing orders and should verify that such information is current and complete.
NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer. NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this specification.
NVIDIA products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.
NVIDIA makes no representation or warranty that products based on these specifications will be suitable for any specified use without further testing or modification. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to ensure the product is suitable and fit for the application planned by customer and to do the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this specification. NVIDIA does not accept any liability related to any default, damage, costs or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this specification, or (ii) customer product designs.
No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this specification. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA. Reproduction of information in this specification is permissible only if reproduction is approved by NVIDIA in writing, is reproduced without alteration, and is accompanied by all associated conditions, limitations, and notices.
ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the NVIDIA terms and conditions of sale for the product.

Trademarks

NVIDIA, the NVIDIA logo, and TensorRT are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2020 NVIDIA Corporation. All rights reserved.