2020 embedded VISION summit®

Practical Guide to Implementing Deep Neural Network Inferencing at the Edge

Toly Kotlarsky
Distinguished Member of Technical Staff, R&D

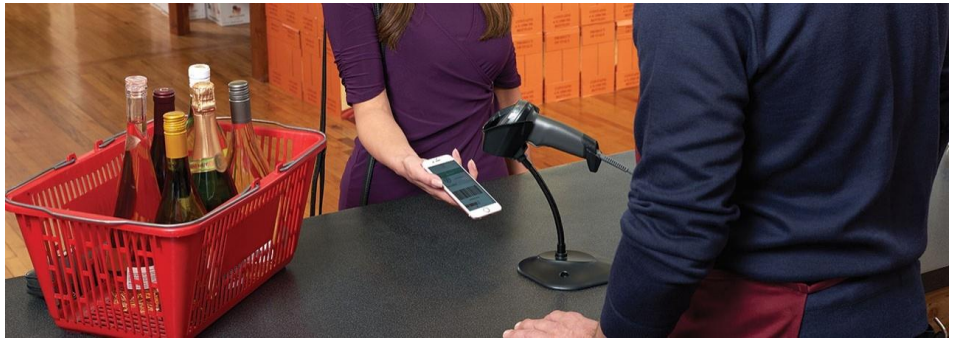
September 2020

ZEBRA

Zebra Technologies

THE EDGE YOU NEED TO SUCCEED

- Zebra empowers those on the front line in retail, healthcare, transportation and logistics, manufacturing and other industries to achieve a performance edge
- As the pioneer at the edge of the enterprise, our products, software, services, analytics and solutions are used to intelligently connect people, assets and data



Why TensorFlow

- TensorFlow is an end-to-end open source platform for machine learning
- Comprehensive, flexible ecosystem of tools, libraries and community resources
- Allows researchers to push the state-of-the-art in ML
- Allows developers to easily build and deploy ML powered applications
    - Image classification
    - Object Detection

Training Deep Neural Network in TensorFlow

IMAGE CLASSIFICATION

Input:

- Labeled dataset of images

Output:

- Frozen graph .pb file (TF-1)
- Saved Model (TF-2)
- TFLITE

- Transfer Learning ("TensorFlow for Poets" example)
- Training from scratch ("Slim" API)

Label	Probability
rabbit	0.07
hamster	0.02
dog	0.91

Training Deep Neural Network in TensorFlow

IMAGE CLASSIFICATION MODELS

- alexnet_v2
- cifarnet
- overfeat
- vgg_a
- vgg_16
- vgg_19
- inception_v1
- inception_v2
- inception_v3
- inception_v4
- inception_resnet_v2
- i3d
- s3dg
- lenet
- resnet_v1_50
- resnet_v1_101
- resnet_v1_152
- resnet_v1_200
- resnet_v2_50
- resnet_v2_101
- resnet_v2_152
- resnet_v2_200
- mobilenet_v1
- mobilenet_v1_075
- mobilenet_v1_050
- mobilenet_v1_025
- mobilenet_v2
- mobilenet_v2_140
- mobilenet_v2_035
- nasnet_cifar
- nasnet_mobile
- nasnet_large
- pnasnet_large
- pnasnet_mobile

Training Deep Neural Network in TensorFlow

OBJECT DETECTION

Input:

- Annotated dataset of images
- CSV files containing information about the images used for model training and testing respectively

Output:

- Frozen graph .pb file (TF-1)
- Saved Model (TF-2)
- TFLITE

- Overview on tensorflow.org
- Tutorial from Edje Electronics

Training Deep Neural Network in TensorFlow

OBJECT DETECTION MODELS

ssd_mobilenet_v1_0.75_depth_coco
ssd_mobilenet_v1_quantized_coco
ssd_mobilenet_v1_0.75_depth_quantized_coco
ssd_mobilenet_v1_ppn_coco
ssd_mobilenet_v1_fpn_coco
ssd_resnet_50_fpn_coco
ssd_mobilenet_v2_coco
ssd_mobilenet_v2_quantized_coco
ssdlite_mobilenet_v2_coco
ssd_inception_v2_coco
faster_rcnn_inception_v2_coco
faster_rcnn_resnet50_coco
faster_rcnn_resnet50_lowproposals_coco
rfcn_resnet101_coco
faster_rcnn_resnet101_coco
faster_rcnn_resnet101_lowproposals_coco
faster_rcnn_inception_resnet_v2_atrous_coco
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco
faster_rcnn_nas
faster_rcnn_nas_lowproposals_coco
mask_rcnn_inception_resnet_v2_atrous_coco
mask_rcnn_inception_v2_coco
mask_rcnn_resnet101_atrous_coco
mask_rcnn_resnet50_atrous_coco

Evaluating Accuracy of the Trained Image Classification Model

- To test and quantify inference accuracy, we developed a Python script ***label_images.py***
- Executes inference on a given dataset of images and produces an accuracy report
- Accuracy score ranges from 0.0 to 1.0, where 1.0 is a perfect score
    - Based on the requested number of top scorers to classify **N** and the min_score
    - Inference on each image gets a score ranging from N to 0, where N is the highest score for the top-1 result, (N-1) is the score for the top-2 result, and so on, 0 is the score for the miss

  Accuracy score = average inference score / N
- Report shows the accuracy score per category and the bottom-line accuracy score

Evaluating Accuracy of the Trained Image Classification Model

```
==== CATEGORY: daisy ====
********
D:\Symbol\Machine Vision\tensorflow-for-poets-2-master\tf_files\flower_photos_train_test\test\daisy\19813618946_93818db7aa_m.jpg

Classification time (1-image) = 0.265 sec

daisy (score=0.95613)
sunflowers (score=0.03515)
classification_score = 3

********
D:\Symbol\Machine Vision\tensorflow-for-poets-2-master\tf_files\flower_photos_train_test\test\daisy\2045022175_ad087f5f60_n.jpg

Classification time (1-image) = 0.255 sec

roses (score=0.55690)
daisy (score=0.39500)
dandelion (score=0.04614)
classification_score = 2
```

Evaluating Accuracy of the Trained Image Classification Model

```
==== RESULTS FOR: tulips ====

Num of images = 50
Average classification time = 0.259 sec
Average score = 0.920 (2.760 out of 3)

Top 1 hit perc = 84.0%
Top 2 hit perc = 92.0%
Top 3 hit perc = 100.0%

============================================================

==== TOTAL RESULTS ====

Num of images = 250
Average classification time = 0.264 sec
Average score = 0.956 (2.868 out of 3)

Top 1 hit perc = 90.8%
Top 2 hit perc = 97.2%
Top 3 hit perc = 98.8%
```

Evaluating Accuracy of the Trained Object Detection Model

- To test and quantify inference accuracy, we developed a Python script ***detect_objects.py***
- Executes inference on a given dataset of images and produces an accuracy report
- Using the given ground-truth CSV file, min_score and IOU threshold, it calculates:
  - Precision, recall, average precision at recall (AP), F_β scores per category
  - Overall mean average precision (mAP) and F_β scores

  F_β scores are calculated for the given set of β values (e.g. 0.5, 1.0, 2.0, etc.)

  where β represents the ratio of how much recall is as important as precision:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

Evaluating Accuracy of the Trained Object Detection Model

```
********
images/test\IMG_2647.JPG

Classification time (1-image) = 0.068 sec

Object # 1 **CORRECT**:
ace
Inference score = 0.99635
Bounding box = (286,87)-(351,301)

Object # 2:
nine
Inference score = 0.98578
Bounding box = (130,83)-(289,311)

Image Results:
  True Positives = 1
  False Positives = 1
  False Negatives = 1
  Precision = 0.500
  Recall = 0.500
```

where “Inference score” is the probability value of the detected class

Evaluating Accuracy of the Trained Object Detection Model

```
==== RESULTS FOR: king ====

True positives = 14
False positives = 1
False negatives = 9
Precision = 0.933
Recall = 0.609
Average true positive score = 0.91492
Min. true positive score = 0.67709
Average precision at recall = 0.630
F(0.50)-score = 0.843
F(1.00)-score = 0.737
F(2.00)-score = 0.654

==== RESULTS FOR: ace ====

True positives = 26
False positives = 8
False negatives = 0
Precision = 0.765
Recall = 1.000
Average true positive score = 0.99662
```

where Average and Min true positive scores are the average and minimum probability values of the detected classes

Evaluating Accuracy of the Trained Object Detection Model

```
============================================================

==== TOTAL RESULTS (min_score = 0.60, iou = 0.50) ====

Num of images = 67
Total true positives = 125
Total false positives = 39
Total false negatives = 18
Overall precision = 0.762
Overall recall = 0.874
Average true positive score = 0.97045
Min. true positive score = 0.67709
Mean average precision (mAP) = 0.760
F(0.50)-score = 0.782
F(1.00)-score = 0.814
F(2.00)-score = 0.849

============================================================
```

Inference at the Edge Using C/C++ Applications

ZEBRA

Raspberry Pi 3 B+

Broadcom BCM2837B0 Cortex-A53 CPU @1.4GHz

$44.95

Setting Up Raspberry Pi for C/C++ Inference

- Make sure that the latest tool-chain is installed in the Raspberry Pi system

  sudo apt-get install build-essential

- Install TensorFlow Lite
  - Download and decompress TensorFlow tar-ball from the git-hub repository
  - Download dependencies

    ./tensorflow/lite/tools/make/download_dependencies.sh

  - Compile static TensorFlow Lite library

    ./tensorflow/lite/tools/make/build_rpi_lib.sh

- Install OpenCV and, possibly, other packages as needed

Setting Up Raspberry Pi for C/C++ Inference

- Convert your trained model to TFLITE format

  python tflite_convert.py …

  - Trained model can be a frozen graph or a Saved Model
  - Can generate a quantized TFLITE file
  - To see all options, execute

  python tflite_convert.py --help

- Use TensorFlow Lite APIs in your C/C++ inference applications
- Compile and run your application in Raspberry Pi

NVIDIA Jetson Nano

Cortex-A57 CPU @1.43GHz + 128-core CUDA GPU

$126.23

Setting Up Jetson Nano for C/C++ Inference

- TensorRT is an NVIDIA's software package that provides support for NVIDIA GPUs in machine learning applications
- UFF is the NVIDIA's format of the neural network models
- PLAN is the TensorRT serialized engine file deployable for inference
- The model conversion tools are pre-installed in Jetson Nano
- In order for trained model to be deployed on NVIDIA's GPUs in C/C++ applications, the model needs to be first converted to the UFF format (currently, works only with TF-1)

python3 convert_to_uff.py frozen_graph.pb -o converted_model.uff [-p config.py]

where optional config.py is a graph pre-processing script, specifying the rules of conversion of layers and operators unknown to TensorRT

Setting Up Jetson Nano for C/C++ Inference

- You can serialize your UFF file and generate PLAN file at run-time in your C/C++ application
- Or, execute trtexec tool off-line to convert the UFF file to the PLAN file

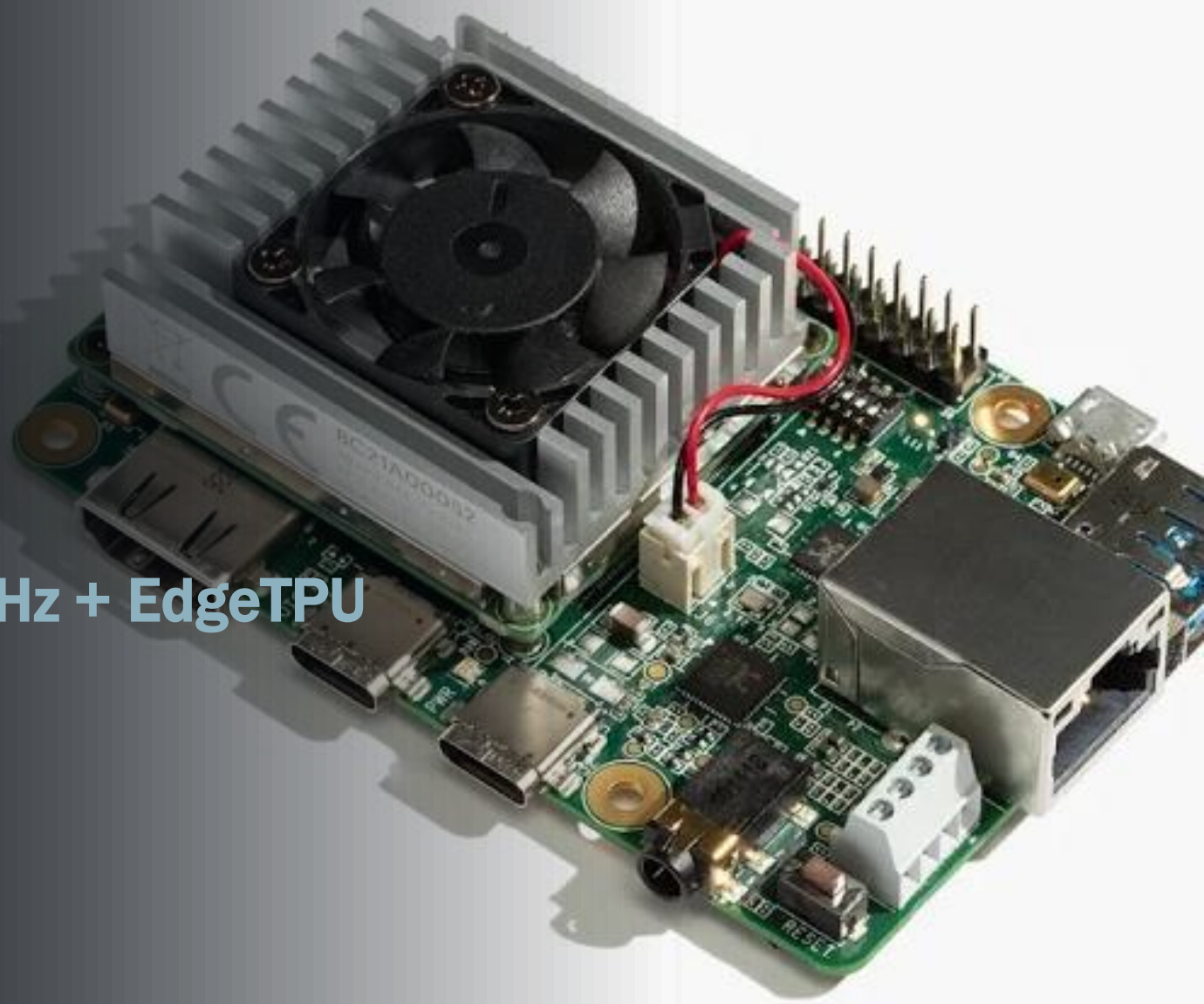
trtexec --uff=converted_model.uff --uffInput=Input,3,300,300 --output=NMS --workspace=100 --saveEngine=trt_engine.plan

- Use TensorRT APIs in your C/C++ inference applications
    - TensorRT run-time libraries and header files are pre-installed in Jetson Nano
- Compile and run your C/C++ application in Jetson Nano

Google Coral Board

NXP i.MX8 Cortex-A53 CPU @1.5GHz + EdgeTPU

$149.99



Setting Up Google Coral Board for C/C++ Inference

- Install and build TensorFlow Lite (similarly to Raspberry Pi)
- Add Edge TPU package repo to your sources
  - Follow instructions in https://coral.ai/software
- Install Edge TPU software packages

  sudo apt-get install edgetpu

  - This is a metapackage that provides both the Edge TPU compiler (edgetpu-compiler) and the Edge TPU runtime (libedgetpu1-std)
- Install Edge TPU header files (edgetpu.h and edgetpu_c.h) for C++ development

  sudo apt-get install libedgetpu-dev

Setting Up Google Coral Board for C/C++ Inference

- Use edgetpu_compiler to convert your trained NN model to the format recognizable by the Edge TPU
  - NN model must be in TFLITE format and it must be quantized
  - After you train and convert your model to TFLITE with quantization, the final step is to compile it with the Edge TPU compiler

  edgetpu_compiler <my_model>.tflite
- I recommend to rename the compiler's output file to have an extension .edgetpu instead of .tflite, so that it could be distinguished from the ordinary TFLITE file
- Use TensorFlow Lite EdgeTPU APIs in your C/C++ inference applications
- Compile and run your C/C++ application in Google Coral Board

C/C++ Inference Application "Ileana"

"Ileana" is a portable application developed by Zebra in C/C++ for testing and benchmarking inference performance of neural network models at the edge on variety of hardware platforms

ZEBRA

Running Inference at the Edge Using “Ileana” Application

- “ileana” can run on any Linux platform
- Can execute inference on a single image file or run real-time inference on images streaming live from a video camera
  - Any UVC/V4L2 compatible camera
- Accepts models in TFLITE, UFF, EdgeTPU formats
  - TFLITE models can run on any Linux platform
  - UFF models can run in Jetson and are automatically serialized to the PLAN format
  - EdgeTPU models can run in Google Coral

Running Inference at the Edge Using “Ileana” Application

Usage:

./ileana [options] [image_file]

Options:

--model=<model file>	
--labels=<label's file>	
--height=<model's image height>	default: 128
--width=<model's image width>	default: 128
--channels=<model’s num of channels>	default: 3
--video=<video source id>	default: 0
--inf_type=<0 for img classif or 1 for obj det>	default: 0
--min_score=<min. score>	default: 0.01
--norm=<image normalization method, 0 or -1>	default: -1 (from -1.00 to +1.00)
--display display image file	default: do not display
--save=<path to save image on 'S' key>	default: do not save

Inference Performance Comparison: Image Classification

DNN model	image size	TFLITE	TFLITE QUANTIZED	TensorRT	EdgeTPU
Raspberry Pi 3 B+					
inception_v3	299 x 299	2038 ms	2197 ms	N/A	N/A
mobilenet_0.50_128	128 x 128	67 ms	43 ms	N/A	N/A
NVIDIA Jetson Nano					
inception_v3	299 x 299	683 ms	851 ms	155 ms	N/A
mobilenet_0.50_128	128 x 128	17 ms	16 ms	18 ms	N/A
Google Coral Board					
inception_v3	299 x 299	1073 ms	1558 ms	N/A	68 ms
mobilenet_0.50_128	128 x 128	43 ms	27 ms	N/A	7 ms

Inference Performance Comparison: Object Detection

DNN model	image size	TFLITE	TFLITE QUANTIZED	TensorRT	EdgeTPU
Raspberry Pi 3 B+					
ssd_inception_v2	300 x 300	1612 ms	1835 ms	N/A	N/A
ssd_mobilenet_v1	300 x 300	658 ms	617 ms	N/A	N/A
NVIDIA Jetson Nano					
ssd_inception_v2	300 x 300	530 ms	692 ms	109 ms	N/A
ssd_mobilenet_v1	300 x 300	188 ms	213 ms	60 ms	N/A
Google Coral Board					
ssd_inception_v2	300 x 300	842 ms	1297 ms	N/A	57 ms
ssd_mobilenet_v1	300 x 300	348 ms	368 ms	N/A	25 ms

In Conclusion…

- Image Classification and Object Detection are the two most popular ML-powered applications for computer vision
- TensorFlow is a great framework for training and evaluating neural network models
- Evaluate and compare accuracy of the models you have chosen and trained for your application prior to their deployment at the edge
- Benchmark inference performance of the models in real hardware platforms
    - Check that inference accuracy in real hardware matches your expectations
    - Find an optimal balance of hardware cost vs. inference performance vs. inference accuracy vs. power consumption, etc. – the balance that works best for your application

Resources

Zebra Technologies Web-Site

https://www.zebra.com

TensorFlow Main Web-Site

https://www.tensorflow.org/

TensorFlow for Poets Tutorial

https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/#0

Image Classification SLIM API

https://github.com/tensorflow/models/tree/master/research/slim

Resources

TensorFlow Object Detection Overview

https://www.tensorflow.org/lite/models/object_detection/overview

Object Detection Tutorial

https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10

Understanding the mAP

https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3

TensorFlow Lite Guide

https://www.tensorflow.org/lite/guide

Resources

TensorRT Developer Guide

https://docs.nvidia.com/deeplearning/sdk/tensorrt-developer-guide/index.html#c_topics

EdgeTPU C++ API

https://coral.ai/docs/edgetpu/tflite-cpp/

2020 Embedded Vision Summit

“Practical Guide to Implementing Deep Neural Network Inferencing at the Edge”

Toly Kotlarsky, May 2020

THANK YOU!

ZEBRA