Making Edge AI Inference Programming Easier and Flexible

Manisha Agrawal
Product Marketing Engineer
September 2020
Agenda

Challenges with edge inference deployment

Open source inference framework advancement

Easier and flexible programming on TI Jacinto™ 7 processors

Programming experience with Jacinto 7 processors compared to desktop
Deploying Deep Learning Model for Edge Inference (1/2)

Training framework

- Caffe
- OpenCV
- ML
- TensorFlow
- Keras
- MxNet

Deploy

Trained model

Edge Inference

- Car
- Wi-Fi
- Surveillance camera
- Robot
Deploying Deep Learning Model for Edge Inference (2/2)

Training framework

- Caffe
- Custom tools
  - Vendor A
  - Vendor B
  - Vendor C
  - Vendor D
  - Vendor E

Proprietary inference framework

- Learn
- Optimizer
- Runtime engine
  - API
  - Interpreter
  - Scheduler
  - Kernel library

Embedded hardware
Edge Inference Programming Challenges

Challenges

- Tools not easy to use
- Unsupported operators
- Accuracy goals not met
- Performance not met
- Power goals not met
Open Source Inference Framework Advancement

Training framework

Caffe

Inference framework

Compiler / Optimizer

Runtime engine

API | Interpreter | Scheduler

Kernel library

Embedded hardware

General purpose

Custom

Open source inference framework

GLOW

ONNX Runtime

TensorFlow Lite

ArmNN

tvm
## Promising Open Source Framework Meeting Majority Customers Need

<table>
<thead>
<tr>
<th>Training framework</th>
<th>Optimizer / Compiler / Graph format</th>
<th>RunTime engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>TensorFlow</td>
<td>TensorFlow Lite format</td>
<td>TFLite RunTime</td>
</tr>
<tr>
<td>ONNX</td>
<td>ONNX format</td>
<td>ONNX-RunTime</td>
</tr>
<tr>
<td>TVM / Amazon SageMaker Neo compiler</td>
<td>TVM / Amazon SageMaker Neo compiler</td>
<td>Neo-AI-DLR</td>
</tr>
</tbody>
</table>
Open Source TFLite RunTime

- Model artifacts
- TFLite RunTime API
- TFLite Kernel library
- General purpose hardware

TFLite RunTime

- Supporting all inference operators on CPU / GPU
Open Source ONNX RunTime

- Model artifacts

ONNX RunTime
- Supporting all inference operators on CPU / GPU

- OpenSource RunTime API
- ONNX Kernel library
- General purpose hardware
Open Source Inference Framework with Hooks for Specialized Hardware

**Compiler & RunTime**

- Supporting all inference operators on CPU / GPU
- Backend for specialized hardware
TI’s First Jacinto™ 7 SoC for Edge Inference

**Accelerating key functions lowers power**
- DSP for computer vision
- Vision processing
- Video, graphics
- Deep learning

**Industry’s most efficient DL architecture**
- Enables passively-cooling designs
- 90% utilization of deep learning accelerator due to smart memory system

**Automotive Quality-ready process technology**
- Power reduction is achieved through smart architecture, not process

https://www.ti.com/product/TDA4VM
NRE-free, royalty-free tools enable high-performance, fixed-point inference on TI processors

**Optimizer**

- Post training quantization for 8-bit and 16-bit
- Range calibration
- Hardware independent optimizations
- Hardware specific optimizations

**RunTime**

- Runtime API
- Deep learning library

TI Deep Learning (TIDL) Inference Framework
TIDL Accelerate All Operators You Rely On

TIDL features

- Accelerates all operators commonly used by CNN vision models on our deep learning accelerator cores
- Out-of-box support for 35+ pre-trained CNN models
- List continues to grow

Popular operators supported include

- Convolution
- Pooling
- Element wise
- Inner-product
- Soft-max
- Bias add
- Concatenate
- Scale
- Batch normalization
- Re-size
- Arg-max
- Slice
- Crop
- Flatten
- Shuffle channel
- Detection output
- Deconvolution/Transpose convolution

Refer to latest Processor SDK user guide document for complete list of accelerated operators and tested models:
https://software-dl.ti.com/jacinto7/esd/processor-sdk-rtos-jacinto7/latest/exports/docs/tidl_j7_01_02_00_09/ti_dl/docs/user_guide_html/md_tidl_layers_info.html
Texas Instruments adopting open source framework with TI Deep Learning (TIDL) Integration

Open source adoption
- Integrating TIDL RunTime in open source RunTime engine
Texas Instruments adopting open source framework with TI Deep Learning (TIDL) Integration

Open source adoption: TVM Compiler

- Integrating TI Optimizer in open source TVM Compiler tool
Now Accelerate All Your Models With Open Source API

- Inference latency in <5 ms for all popular classification models
- Minimal accuracy loss with Post Training Quantization and Quantization Aware Training tools from TI

Run any model from open source inference frameworks on TI processors!

TensorFlow Lite models
ONNX models
TVM/Amazon SageMaker
Neo compiler

TFLite RunTime
ONNX-RunTime
Neo-AI-DLR

User Application
Python / C / C++

TFLite/ONNX-RT/Neo-AI-DLR
API | interpreter | scheduler

TIDL RunTime
Linux OS
CPU

Jacinto 7 processor

C7x DSP with MMA*
Deep learning accelerator

*MMA: Matrix Multiplication Accelerator (Tensor Processing Unit)
Your DL programming experience on Jacinto 7 processors is the same as desktop computer programming.

This includes -

• Open source LINUX callable APIs in Python / C / C++ between PC and target board
• Jupyter Notebook examples

Download directly from ti.com

Programming Example: Amazon SageMaker Neo & Neo-AI-DLR

Model compilation

Location of model artifacts
Amazon SageMaker needs the path to the model artifacts in Amazon S3. To find the path, look in your Amazon S3 directories.

```
s3://sagemaker-ti-test/mobilenet_v2_1_0_224_frozen.tgz
```

To find a path, go to Amazon S3

Data input configuration
Amazon SageMaker needs to know what the shape of the data matrix is.

```
{'input': [4, 224, 224, 3]}
```

Machine learning framework
Choose the machine learning framework that your model was trained in.

```
TensorFlow
```

Target device
Amazon SageMaker needs to know where you intend to deploy your model: to an Amazon SageMaker ML instance or to an AWS IoT Greengrass device.

```
TDA4VM
```

S3 Output location
Amazon SageMaker needs the path to the S3 bucket or folder where you want to store the compiled module.

```
s3://sagemaker-ti-test
```

SageMaker Neo console

Provide model information

Select TDA4VM as target device

RunTime

Create RunTime

```
from dlr import DLModel
import numpy as np
import cv2
from pathlib import Path

MODEL_PATH = Path("/.../build/ssd_mobilenet_v2").resolve()
DATA_PATH = Path("/.../build/street_small.npy").resolve()

def test_ssd_mobilenet_v2_model():
    model = DLModel(MODEL_PATH.as_posix())
    data = np.load(DATA_PATH)
    assert model.get_input_names() == ['image_tensor']
    assert model.get_output_names() == ['detection_scores', 'detection_classes', 'num_detections']
    assert model.get_input_types() == ['uint8']
    assert model.get_output_dtypes() == [np.float32, np.float32, np.float32]
    outputs = model.run(['image_tensor': data])
    assert outputs[0].shape == (1, 100, 4)
    assert outputs[1].shape == (1, 100)
    assert outputs[2].shape == (1, 100)
    detections = np.multiply(outputs[1], outputs[2])
    expected = np.zeros(detections.shape)
    expected[:16] = np.array([1., 1., 1., 2., 3., 1.])
    comparison = detections == expected
    assert comparison.all()

if __name__ == '__main__':
    test_ssd_mobilenet_v2_model()
```

Run inference

Output

Provide output location

Compile the model

© 2020 Texas Instruments
def infer_tflite_model():
    interpreter = tf.lite.Interpreter(model_path=args.model_path)
    interpreter.allocate_tensors()

    input_details = interpreter.get_input_details()
    output_details = interpreter.get_output_details()

    print(output_details)
    # check the type of the input tensor
    floating_model = input_details[0]['dtype'] == np.float32

    # N x H x W x C, H:1, W:2
    height = input_details[0]['shape'][1]
    width = input_details[0]['shape'][2]
    img = Image.open(args.input_file).resize((width, height))

    # add N dim
    input_data = np.expand_dims(img, axis=0)

    if floating_model:
        input_data = (np.float32(input_data) - args.input_mean) / args.input_std

    interpreter.set_tensor(input_details[0]['index'], input_data)
    tidl_delegate = TfLiteTIDLDelegateCreate()
    interpreter.ModifyGraphWithDelegate(tidl_delegate)
    interpreter.invoke()

    output_data = interpreter.get_tensor(output_details[0]['index'])
    results = np.squeeze(output_data)
    top_k = results.argsort()[-5:][::-1]
    print(top_k)

    infer_tflite_model()
Deep Learning System Performance on Jacinto-7 TDA4VM: Example demo

5 simultaneous deep learning algorithms on 3x 1MP camera each @ ~16 fps
- Parking spot detection
- Vehicle detection
- Semantic segmentation
- Motion segmentation
- Depth estimation

Inference resolution: 3x768x384

<table>
<thead>
<tr>
<th>Camera 1</th>
<th>Camera 2</th>
<th>Camera 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantic segmentation</td>
<td>Parking spot detection</td>
<td>Vehicle detection</td>
</tr>
</tbody>
</table>

Resource loading

| | A72: 6% | C7x+MMA: 94% |
|---------------------------|
| DDR BW: | 26% |

Inference resolution: 3x768x384
Key Takeaways: Deep Learning Edge Inference at TI

- **Easier to use**
  - Opensource Linux callable RunTime APIs supported to program SoC
  - Embedded development environment same as a desktop computer environment
- **More flexible**
  - Supports TFLite, ONNX-RT or TVM/Neo-AI-DLR
  - Supports compilation at the edge or in the cloud with Amazon SageMaker Neo
- **Provide wide model coverage**
  - All TFLite, ONNX, TVM and SageMaker Neo models supported on TI SoCs
- **High compute performance, high throughput, low latency and low power consumption**
  - Accelerates the model on TI’s deep learning accelerator C7x+MMA to provide the best combination of system power, deep learning performance and latency at the edge

Jacinto 7 processor silicon is available today!
Explore Deep Learning With TI Today

Full development
TDA4 EVM
http://www.ti.com/tool/TDA4VMXEVM

Turn-key designs
Automotive version of TDA4V Mid
http://www.ti.com/tool/D3-3P-TDAX-DK

Software development kits
TI Processor SDK – Seamlessly reuse and migrate Linux, Linux-RT and TI-RTOS software across TI processors

Support
https://e2e.ti.com

Open source RunTime engines with TIDL acceleration is packaged as part of TI’s NRE-free, royalty-free Processor SDK!
Thank You!