Embedded low-power deep learning with TIDL

Manu Mathew
Principal Engineer &
Member Group Technical Staff

Kumar Desappan
Member Group Technical Staff

Pramod Kumar Swami
Principal Engineer &
Member Group Technical Staff

Soyeb Nagori
Senior Principal Engineer &
Senior Member Technical Staff

Biju Moothedath Gopinath
Engineering Manager

Automotive Processors
Texas Instruments

Introduction

Computer-vision algorithms used to be quite different from one another. For example, one algorithm would use Hough transforms to detect lines and circles, whereas detecting objects of interest in images would require another technique such as histograms of oriented gradients, while semantic segmentation would require yet a third type of algorithm.

Deep learning methods, including convolutional neural networks (CNNs), have revolutionized machine intelligence, helping algorithms become more accurate, versatile and autonomous. Deep learning has also revolutionized automotive applications. Many state-of-the-art algorithms for advanced driver assistance systems (ADAS) now require deep learning methods, including the detection of lane markings and the detection and classification of various objects such as pedestrians, vehicles, cyclists and traffic signs. Deep learning has emerged as a key technology that provides the best accuracy for most of these algorithms. The tools described in this paper help enable ADAS algorithms on automotive processors from Texas Instruments (TI).

Deep learning provides a systematic way to enable a variety of algorithms. For example, deep learning configurations for many algorithms operate quite similarly, making deep learning a perfect candidate to accelerate processing speeds through software- and hardware-optimization techniques. In this paper, we will specifically address software optimization: highly optimized software components that maximize the efficiency of the available hardware and can increase the speed at which deep learning algorithms run. Algorithmic optimization involves developing faster algorithms to achieve the same end result faster or better. Providing libraries and components that are easy to use and integrate into existing system frameworks improve time to market. These are the goals of the tools we'll describe in this paper.

TI's Jacinto™ TDA2, TDA2P and TDA3 automotive processors enable the processing and fusing of data from camera, radar and ultrasonic sensors to support ADAS functionality[1]. These sensors enable object detection, classification and tracking algorithms for automatic emergency braking or driver monitoring, as well as stitching multiple camera streams together for surround views. Other algorithms include lane detection for lane-keep assist and the detection of 3-D structures for

parking assist. These processors can also perform semantic segmentation, which can help identify the free space available for driving by classifying which pixels of an image belong to the road and which pixels do not.

TI deep learning (TIDL) is a suite of components that enables deep learning on TI embedded devices. TIDL has a highly optimized set of deep learning primitives that provide the best accuracy, speed and memory usage trade-offs. It also provides an easy way to use a model from one of the popular deep-learning training frameworks and run it on a TDA-based embedded platform very quickly. Ease of use and high performance are the two key motivations behind TIDL.

Figure 1 illustrates the TIDL suite of components. The first part of the development flow is for training a network model and is best accomplished within popular training frameworks. The next step is using the TIDL device translator tool to convert network models into an internal format best suited for use inside the TIDL library. The final step is to run the converted network model on the embedded TDA device using TIDL-provided application programming interfaces (APIs).

TIDL can run full-frame CNNs, which some of the ADAS algorithms, such as object detection and semantic segmentation, require. TIDL can also run object-classification algorithms that operate on a small region of interest in the image.

Deep learning for low-power devices

Deep learning involves training and inference. Training usually occurs offline using a large data set on servers or PCs with external graphics processing units (GPUs). Real-time performance or power is not an issue during this phase. However, during actual inference, when a low-power device executes an algorithm such as lane detection, real-time performance and power consumption are important. Several publicly available deep learning frameworks enable the training of CNN or other deep learning models. Popular frameworks include Caffe, TensorFlow, CNTK, MxNet and PyTorch.

Most of these platforms are optimized for central processing units (CPUs) or GPUs and run at very high speeds, especially on the GPUs. However, there is a lack of support for low-power embedded devices such as digital signal processors (DSPs). Because DSPs consume much less power than GPUs, systems using DSP processors can be placed in small cases that provide limited thermal dissipation or in portable devices that have limited battery power.

TI developed the TIDL suite of components in order to address the gap for supported DSPs. TIDL does not address the training of deep-learning models, which the popular deep-learning frameworks can best handle. Instead, TIDL addresses the inference part of deep learning, using a trained model from a supported network and running it at a very high speed on a

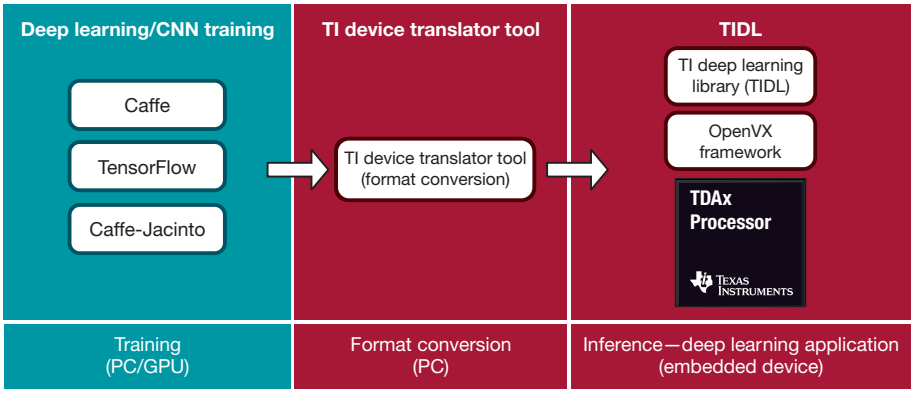

***Figure 1**. TIDL development flow.*

supported low-power embedded processor like one from the TI TDA family.

The TI device translator tool enables development on open frameworks and provides push-button PC-to-embedded porting. TIDL abstracts embedded development, provides high-efficiency implementation and is platform scalable.

Features

As we discussed, the purpose of TIDL is to enable ease of use and provide optimized inference. Ease of use is achieved by providing a way to use the trained network models in the TIDL library. Thus, one primary feature is that TIDL can understand the trained output of popular frameworks.

TIDL has achieved optimized inference through software optimizations that enable it to use the underlying hardware resources optimally and through algorithmic simplifications, such as sparse convolutions that reduce the number of operations required for CNN.

TIDL also offers these features:

- **Layer types**. Deep-learning models such as CNNs are arranged in layers. A layer typically consists of certain mathematical operations such as filters, rectification linear unit (ReLU) operations, downsampling operations (usually called average pooling, max pooling or striding), elementwise additions, concatenations, batch normalization and fully connected matrix multiplications. TIDL supports most of the popular CNN layers present in frameworks such as Caffe and TensorFlow.
- **Sparse convolution**. A convolution algorithm that takes advantage of zero coefficients and runs faster when a significant portion of the weights are zero is called sparse convolution. TIDL uses an efficient convolution algorithm that runs faster when using sparse models. Speed-up can be quite significant when sparsity is high.
- **Quantized inference and on-the-fly quantization**. The trained model is a floating-point model. However, floating point is not the best for execution speed on low-power embedded devices. Thus, it is important to convert the floating-point model such that inference execution can use fixed-point operations (with example convolutions done using 8-bit or 16-bit integer multiplications). TIDL and its device translator tool will automatically convert floating point to fixed point so that the training algorithm or framework does not need to do anything special for fixed-point inference in TIDL. This is called on-the-fly quantization, a sophisticated feature that increases execution speed significantly and takes care of varying input-signal characteristics and intermediate layer outputs. TIDL supports both 8-bit and 16-bit quantization. The drop in accuracy due to quantization is small for several popular networks.
- **Host emulation**. While TIDL actually runs on embedded devices, host emulation mode enables you to perform a sanity check. In host emulation mode, TIDL runs on the host PC, emulating each of the CNN network layers and producing the expected output. Thus, you can check the expected output on the device without actually using an embedded device.
- **Support for a variety of training frameworks**. The TIDL device conversion tool is compatible with trained models from BVLC/Caffe, TensorFlow, NVIDIA/Caffe and TIDSP/Caffe-Jacinto. Each of these tools has their own strengths; you can choose the one that suits your requirements.

- **Low power consumption**. Full-frame semantic segmentation at 15 fps consumes only 2.5W of computing power on the TDA2x system-on-chip (SoC).

Sparse convolution

The complexity of the overall network should be restricted such that it fits well within the computing capability of the targeted device. Typically, the convolution layers are the most computationally intense and will determine how fast the inference runs—so it is important to reduce the complexity of convolution layers. TIDL supports sparse convolution, which can execute the inference much faster when there are a lot of zero coefficients.

Using sparse convolution algorithms eliminates the need for multiplications whenever the weights are zeros. Sparse training methods can induce 80 percent or more sparsity in most convolution layers —in other words, making 80 percent of the convolution weights zero. At TI, we have observed a 4× execution speed increase when nearly 80 percent of the weights in the convolution layers are zeros. Sparsity is optional, however, and TIDL can work with conventional non-sparse models as well.

Training

Caffe and TensorFlow are the currently supported training frameworks; in other words, TIDL can import models trained in these frameworks using the device translator tool. As we mentioned, TIDL also supports various Caffe flavors including BVLC/Caffe, NVIDIA/Caffe and TIDSP/Caffe-Jacinto.

Caffe-Jacinto is a custom fork of Caffe that provides tools to train models with sparsity. Caffe-Jacinto models help you get started on training with sparsity and include detailed documentation about how to proceed with the training.

Forcing convolution weights to zero can reduce the accuracy of the deployed algorithm. For example, you will want to avoid accuracy drops of 25 percent (that should be within 1 or 2 percent) for a trained network model for image classification when introducing sparsity. Training models with sparsity (sparsification) without losing accuracy significantly are an important aspect of the training phase. Reference [2] offers additional details about training with sparsification.

Sparsification at training time is useful only if the inference framework (in our case TIDL) is capable of performing sparse convolutions efficiently. Caffe-Jacinto is a good training framework for generating sparse models that can run much faster in TIDL.

Device converter tool

Training can be done in floating point. Conversion from floating-point to fixed-point models happens on the fly inside the device converter tool and TIDL. This method provides the maximum ease of use, because you can proceed with the training without any concerns regarding quantization.

Results

References [3] and [4] are demonstrations of TIDL used for real-time semantic segmentation on TDA2 automotive processors. **Figure 2** is a sample frame that shows semantic-segmentation output in a colorful way. The purple color shows pixels classified as road, blue shows pixels classified as vehicles, red shows pixels classified as pedestrians and cyclists, and yellow would show pixels classified as road signs (not shown in the figure).

***Figure 2**. Semantic segmentation using TIDL on a TDA2 SoC.*

Inference method	Configuration for inference	Giga multiply accumulations per second (MACs)	Giga cycles	Time (ms)	Frames per second (fps)
Dense	JSegNet21 nonsparse	8.843	0.700	194.44	5.14
Sparse	JSegNet21 sparse (80%)	1.540	0.188	52.22	20.22

Table 1. *Measurements from the TDA2x SoC for inferring semantic segmentation of an image with 1,024-by-512 pixels resolution.*

It is seen from experiments that the classification accuracy drop for a typical CNN network is around 1 percent, while inducing 80 percent sparsity. The total drop in accuracy due to sparsification and quantization is within 2 percent. The same observation is true for semantic segmentation as well. Further details on CNN network structures and accuracy are available in [2] and [5].

Table 1 lists the results of a semantic-segmentation network running on the TDA2 SoC using TIDL. As you can see, inducing around 80 percent sparsity increases the speed of inference from about 5 fps to about 20 fps for 1,024-by-512 pixels full-frame semantic-segmentation applications.

How to choose your network configuration

While popular networks can run on TIDL, low-power embedded devices do not have the same level of computing capability as high-power (but costly) GPUs. The network deployed must fit within the capability of the device. This will vary depending on the embedded device.

Algorithm developers sometimes look at the model size (the number of parameters in the model) to determine inference complexity, but that's a small issue in automotive applications. Inference complexity depends on several factors, including the number of multiplications, data-transfer requirements for input/output activations and the transfer of weights. For models like residential networks (ResNets) that do not have heavy, fully connected layers, the number of multiplications required for inference of a certain-sized image is often a good indicator of complexity.

You can also look at the examples given in Caffe-Jacinto models to understand the networks suitable for inference on TDA2x devices. As the computing capability for CNN increases, future TI ADAS SoCs will likely run much more complex models.

How to obtain TIDL

TIDL is part of TI's processor software development (SDK) for vision, which provides an out-of-the-box demo of deep-learning-based semantic segmentation. In the vision SDK, you'll find TIDL at <VSDK>\ti_components\algorithms_codecs [6].

The TIDL package offers detailed documentation on how to use it, the performance of different layers, example networks to demonstrate translation and inference, and other relevant information. It is supported on both embedded vision engine (EVE) and C66x DSP cores on TDA2, TDA2P and TDA3 devices and also comes with a standalone test bench for you to execute and measure the performance of your network without having to understand other system complexities.

References

1. Jacinto TDAx ADAS SoCs, with heterogeneous hardware and software architecture for ADAS.
2. "Sparse, Quantized, Full Frame CNN for Low Power Embedded Devices," which focuses on the training portion of the whole process

and goes into detail regarding the inclusion of sparsity.

3. "TI's Deep Learning-Based Semantic Segmentation on TDA Processors."
4. "Texas Instruments Demonstration of Deep Learning-Based Semantic Segmentation."
5. "Caffe-Jacinto – embedded deep learning framework."
6. "TI Vision SDK, Optimized Vision Libraries for ADAS Systems."

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265