Excel@FIT 2023
http://excel.fit.vutbr.cz

Converter between formats of Deep Neural Network models on mobile platforms

Martin Pavella*

Abstract
One of the most popular formats for representing *Deep Neural Network* (DNN) models is *ONNX*. The development of drivers for HW accelerators on embeded systems is expensive and *ONNX* is rarely supported. The necessary SW support is typically only implemented for the *TensorFlow Lite* (*TFLite*) DNN model format. Currently the options for conversion of pre-trained *ONNX* models to *TFLite* are inadequate and produce sub-optimal models.
The result of my work is a direct converter of *ONNX* models to *TFLite* which focuses on producing as efficient models as possible. The program was verified on DNN models for image classification, object detection, segmentation and acoustic data analysis. The converted models produce identical outputs as the original ones and experiments show a significant improvement in inference speed.

*xpavel39@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

ONNX is a popular format for representing Deep Neural Network (DNN) models. HW accelerators for inference on mobile and embeded systems are however typically only supported by models in the *TensorFlow Lite* (*TFLite*) format.

A DNN model can often be represented in either format. Converting a pre-trained *ONNX* model to *TFLite* can provide a significant increase in inference speed as well as a reduction in model size.

Current solutions[1] for such conversion first convert the *ONNX* model to the *TensorFlow* format and then apply the *TensorFlow* to *TFLite* converter[2] by *Google*. This approach often introduces unwanted artifacts and restrictions for tensor quantisation.

My solution is a direct converter of *ONNX* models to *TFLite*. It bypasses the *TensorFlow* to *TFLite* converter, which allows for total control over the resulting model. This approach generates efficient models which produce identical outputs as the original ones. The improved utilization of HW accelerators has significantly increased the inference speed of models after conversion.

The program was developed and tested in collaboration with the *NXP* company.

2. The need for conversion

Many frameworks for training and inference of DNN models use the *ONNX* format, as it is widely supported and open source. Special hardware accelerators called *Neural Processing Units* (NPU) are being designed to accelerate costly operations such as matrix multiplication or convolution on mobile and embeded devices. The development of drivers and supporting SW for them is very expensive and typically a given NPU is only supported by *TFLite*. As these accelerators can significantly increase the speed of model inference, the motivation to convert *ONNX* models to the *TFLite* format is strong.

Current solutions take a shortcut by first converting to the *TensorFlow* format and then using the existing converter to *TFLite* by *Google*. This approach takes away the ability to precisely control the output model, which sometimes results in sub-optimal models. For instance *ONNX* uses two operators to represent depth-wise convolution while *TFLite* defines a single dedicated operator. The converters often

[1] For example: `https://github.com/PINTO0309/onnx2tf` or `https://github.com/onnx/onnx-tensorflow`

[2] `https://www.tensorflow.org/lite/models/convert/convert_models`

avoid this optimization. Sometimes extra operators are added to transform tensors dynamically during inference even when static transformation during conversion is possible.

3. Proposed solution

My approach was to design a direct converter from *ONNX* to *TFLite* which provides total control over the output model. It works by deserializing the input *ONNX* model and representing it as a hierarchy of objects. A part of this hierarchy is a directed computational graph, where each node represents an operator. As suggested on **Fig. 1**, this graph is analyzed for patterns of operators which are then converted and added to the newly forming *TFLite* model. *TFLite* uses a directed computational graph to represent its DNN as well, but it defines its own set of operators with different behavior. The conversion between operators of these formats is the main challenge of model conversion. The operators have complex behavior and edge cases are difficult to identify and implement. Operator conversion is a complicated and evolving problem as new operators are constantly updated and new ones are being introduced.

Both formats use different ways to represent tensors. *ONNX* uses NCHW while *TFLite* only supports NHWC. These formats define how tensor dimensions are stored in memory. Static tensors must be transformed during conversion and dynamic tensors sometimes need to be reshaped using operators.

4. Results

The converter has been validated on models used for classification, object detection, segmentation and analysis of acoustic data.

The example on **Fig. 2** shows a selected portion of the *Alexnet* model. It is a convolutional DNN for image classification [1] and can be relatively efficiently represented in *TFLite*. The *TFLite* model has a *Transpose* operator, which is not present in the original *ONNX* version. This is because the following *Reshape* operator flattens the tensor into a 2D matrix. In the *ONNX* model, the tensor is flattened from the NCHW format. Flattening a tensor from a different format would cause its values to be at different positions in the matrix, which would cause undefined behavior of the following operators.

The files with *TFLite* models use a binary *flatbuffer* format, which can result in smaller size then in the case of *ONNX*, which uses *protocol buffers*. However the need to add extra operators often cancels out the savings. In the case of an efficiently converted model, the file size has been reduced by up to 420 kB.

5. Impact on inference

The converted models produce nearly identical outputs as their *ONNX* versions. The absolute errors tend to be in the range of 10^{-6} to 10^{-12} depending on the model, which can be attributed to accumulated errors of floating point calculations.

With the help of the NXP company and their specialized HW, we were able to test the effects of model conversion on inference speeds on target platforms. **Tab. 1** shows a significant improvement in the time it takes to complete inference of selected models. The *TFLite* variants display up to 30% faster inference speeds and the effects are most noticeable when a smaller number of threads is used.

6. Limitations

ONNX supports over 140 operators. Implementing the conversion of all of them is far beyond the scope of this work. The aim was to focus on a smaller subset of operators which are used by many models. So the introduction of a new model will occasionally require implementation of the conversion of new operators.

ONNX is a more complex format which defines various data structures and powerful operators. As a result, not every model can be represented using *TFLite's* limited expressive options. In some cases it is possible to convert a model, but it requires the addition of extra operators. This can result in a suboptimal model, which sometimes defeats the purpose of conversion.

Acknowledgements

I would like to thank Ing. Róbert Kalmár of NXP for his professional experience which was the source for section 2.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.