Deep Learning In OpenCV

Wu Zhiwen
zhiwen.wu@intel.com
Agenda

- Background information
- OpenCV DNN module
- OpenCL acceleration
- Vulkan backend
- Sample
What is OpenCV?

- Open Source Compute Vision (OpenCV) library
- 2500+ Optimized algorithms for compute vision and machine learning
- C/C++/Python compatible and cross platform
- 20000+ forks in GitHub
- OpenCV 4.0 is on the way
  - Switch to C++ 11
  - No longer binary-compatibility
  - Better performance on CPU (AVX2)
  - Compact footprint
  - Big revision of DNN module
Key concepts of Deep Neural Networks (DNN)

- Node/Layer/Network/Deep Neural Networks

- Node/Neuron/Perceptron

- Layer

- Deep Neural Network
Key concepts of Deep Neural Networks (DNN)

- Training
  - step1: initialize weights
  - step2: set input data (e.g. an image) and compute the output
  - step3: compare the output and the ground truth and calculate the error
  - step4: modify the weights and go to step 2 until the error is small enough

Complicated?
Deep Learning Frameworks will do that for you
Key concepts of Deep Neural Networks (DNN)

- **Inference/forward/predict**
  You have a trained model, i.e. weights and other parameters.
  Set input data and use Deep Learning Framework to compute the output.

- **Use case**
OpenCV DNN module

- Included in main OpenCV repo since version 3.3
- Inference only
- Compatible to many popular Deep Learning frameworks
OpenCV DNN module

Why we need a new wheel of DNN in OpenCV?

- **Lightness**
  - inference only can simply the code, speed up the installation and compilation process

- **Convenience**
  - build-in implementation, minimum external dependency
  - easy to add deep networks support to your existed OpenCV project

- **Universality**
  - Unified interface to manipulate net models
  - Support multiple target device and OS
    - Device: CPU, GPU, VPU
    - OS: Linux, Windows, Android, MacOS
OpenCV DNN module

- Support ~40 layer types

- AbsVal
- AveragePooling
- BatchNormalization
- Concatenation
- Convolution (including dilated convolution)
- Crop
- Deconvolution, a.k.a. transposed convolution or full convolution
- DetectionOutput (SSD-specific layer)
- Dropout
- Eltwise (+, *, max)
- Flatten
- FullyConnected
- LRN
- LSTM
- MaxPooling
- MaxUnpooling
- MVN
- NormalizeBBox (SSD-specific layer)
- Padding
- Permute
- Power
- PReLU (including ChannelPReLU with channel-specific slopes)
- PriorBox (SSD-specific layer)
- ReLU
- RNN
- Scale
- Shift
- Sigmoid
- Slice
- Softmax
- Split
- TanH
OpenCV DNN module

- Network well tested
  - AlexNet
  - GoogLeNet v1 (also referred to as Inception-5h)
  - ResNet-34/50/...
  - SqueezeNet v1.1
  - VGG-based FCN (semantical segmentation network)
  - ENet (lightweight semantical segmentation network)
  - VGG-based SSD (object detection network)
  - MobileNet-based SSD (light-weight object detection network)
Architecture of DNN module

Language Bindings
(Python, Java)

Accuracy test, Perf test, Samples

Top-level C++ API: Load a net model, run it, retrieve network outputs

Implementation level: model importers, DNN engine, layer implementations,

Acceleration Layer

SSE, AVX, parallel_for (CPU)

OpenCL (GPU)

Halide (CPU, GPU)

Intel IE (clDNN, MKL-DNN, Movidius VPU)
### Backend and target

**Acceleration Layer**

- **SSE, AVX, parallel_for (CPU)**
- **OpenCL (GPU)**
- **Halide (CPU, GPU)**
- **Intel IE (cLDNN, MKL-DNN, Movidius VPU)**

**E.g. use the build-in GPU acceleration**

```python
net.setPreferableBackend(DNN_BACKEND_OPENCV);
net.setPreferableTarget(DNN_TARGET_OPENCL);
```
Network optimizations

- DNN module implemented its own framework internally, these optimizations are not tied to any specific Deep Learning Frameworks.

- Benefit all the net models no matter what their original framework is.
Layer Fusion

DNN module analysis network structure and, if possible, merge some layers into another one.

structure in ResNet50
Layer Fusion

Structure in ResNet50
Layer Fusion

structure in SSD
Memory reuse

- memory usage without reuse
Memory reuse

- Reuse input memory
- Reuse previously allocated memory

Data Flow

Layer1

Layer2 (in-place)

Layer3

allocated memory

reference memory

Reuse input Mem

General reuse algorithm

Reference
OpenCL acceleration

- Enable OpenCL acceleration

```python
net.setPreferableBackend(DNN_BACKEND_OPENCV);
net.setPreferableTarget(DNN_TARGET_OPENCL);
```

Choose FP16:

```python
net.setPreferableTarget(DNN_TARGET_OPENCL_FP16);
```

- No external dependency except OpenCL runtime
- Support FP 32 and FP16 data format
OpenCL acceleration

- Highly optimized convolution kernels
  - auto-tuning to find the best kernel configurations for a specific deployment environment
  - A set of pre-tuned kernel configurations built in the library
  - Tuning your own convolution kernel
    If you want to get the best performance for your specific deployment, try to run auto-tuning instead of using the default configurations.
  - How to enable auto-tuning?
    
    "export OPENCV_OCL4DNN_CONFIG_PATH=/path/to/config/dir"
    
    If you enable auto-tuning, the first time running a net model will be a little bit long. Next time, DNN module will use the cached configs directly and no need tuning again.
OpenCL acceleration

- For better performance on Intel GPU, use Neo driver if possible
  - Neo is the open-source OpenCL driver for Intel GPU
  - Supported Platforms

  Intel Core Processors with Gen8 graphics devices (formerly Broadwell) - OpenCL 2.1
  Intel Core Processors with Gen9 graphics devices (formerly Skylake, Kaby Lake, Coffee Lake) - OpenCL 2.1
  Intel Atom Processors with Gen9 graphics devices (formerly Apollo Lake, Gemini Lake) - OpenCL 1.2

- Use the version as new as possible
  new version always has better performance
OpenCL acceleration

- Performance Data (in milliseconds):

<table>
<thead>
<tr>
<th>Model</th>
<th>DNN, C++</th>
<th>DNN, OpenCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>19.32</td>
<td>11.83</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>23.08</td>
<td>8.20</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>53.26</td>
<td>15.74</td>
</tr>
<tr>
<td>SqueezeNet V1.1</td>
<td>5.94</td>
<td>2.60</td>
</tr>
<tr>
<td>Inception-5h</td>
<td>24.30</td>
<td>9.27</td>
</tr>
<tr>
<td>Enet @ 512*256</td>
<td>68.26</td>
<td>17.26</td>
</tr>
<tr>
<td>OpenFace(nn4.small2)</td>
<td>17.47</td>
<td>4.02</td>
</tr>
<tr>
<td>MobileNet-SSD @ 300*300 20 classes Caffe</td>
<td>30.89</td>
<td>8.71</td>
</tr>
<tr>
<td>MobileNet-SSD v2@ 300*300 90 classes, TensorFlow</td>
<td>47.57</td>
<td>15.40</td>
</tr>
</tbody>
</table>

**Configuration:**
- **OS:** Linux 4.16.0 x86_64 (Ubuntu 16.04)
- **Compiler:** c++ 5.4.0
- **OpenCV:** 3.4.3-308-g761c269
- **CPU:** Intel(R) Core(TM) i7-6770HQ CPU@2.60GHz x8
- **GPU:** Intel® Iris™ Pro Graphics 580 (Skylake GT4e)

For more performance data, see: https://github.com/opencv/opencv/wiki/DNN-Efficiency
Vulkan backend

- Vulkan is the next generation Graphics and Compute API from Khronos, the same cross-industry group that maintains OpenGL
- Extend the usage of GPU acceleration for DNN module
- Use compute shader to implement layer computation
Vulkan backend

- A PR for Vulkan backend is in review

https://github.com/opencv/opencv/pull/12703

**dnn: Add a Vulkan based backend #12703**

wzw-intel wants to merge 5 commits into opencv/master from wzw-intel/vkcom

wzw-intel commented 12 days ago • edited

This commit adds a new backend "DNN_BACKEND.VKCOM" and a new target "DNN_TARGET.VULKAN". VKCOM means vulkan based computation library.

This backend uses Vulkan API and SPIR-V shaders to do the inference computation for layers. The layer types that implemented in DNN_BACKEND.VKCOM include: Conv, Concat, ReLU, LRN, PriorBox, Softmax, MaxPooling, AvePooling, Permute

This is just a beginning work for Vulkan in OpenCV DNN, more layer types will be supported and performance tuning is on the way.
Sample: real-time objection detection with MobileNet-SSD

More samples at: https://github.com/opencv/opencv/tree/master/samples/dnn
Q & A
Backups
OpenCL acceleration

- Auto-tuning
  - For each convolution “key”, generate a set of kernel configurations
  - Compile kernel for each kernel configuration, run kernel, get running time
  - Choose the best kernel configuration and store it on disk or memory

```
input_blob_shape: (0, 3, 300, 300)
output_channel: 64
filter_size: (3, 3)
stride_size: (2, 2)
dilation_size: (1, 1)
padding_size: (1, 1)
group: 1
has_bias: 1
activation_type: 0
eltwise: 1
half_float: 1
eu: 72
```

A convolution “key” is a combination of all convolution parameters and GPU's execution unit number.

A kernel_config is a combination of tile size, simd size and kernel type.

```
a set of kernel_config
(tile_h, tile_w, simd_size, kernel_type):
(2, 32, 8, 2),
(1, 32, 16, 2),
(4, 4, 8, 5),
(4, 4, 16, 5)
.....
```

Best kernel config: (1, 32, 16, 2)
Key concepts of Deep Neural Networks (DNN)

- A sample: GoogLeNet-V1

21 convolution layers + FC layer