June, 17th, CVPR

Sergei Nosov, Deep Learning Manager

IOTG Computer Vision (ICV)

Internet of Things Group
I sometimes see people refer to neural networks as just “another tool in your machine learning toolbox”. They have some pros and cons, they work here or there, and sometimes you can use them to win Kaggle competitions. Unfortunately, this interpretation completely misses the forest for the trees. Neural networks are not just another classifier, they represent the beginning of a fundamental shift in how we write software. They are Software 2.0.

-- Andrej Karpathy, Director of AI at Tesla. Previously Research Scientist at OpenAI and PhD student at Stanford.
Software 1.0 vs Software 2.0

• In Software 1.0 with each line of code the programmer identifies a specific point in program space with desired behavior.

• In Software 2.0 the program is written in more abstract language (weights of the network). Instead of writing it directly we specify the desired behavior and the program is found by optimization.

https://medium.com/@karpathy/software-2-0-a64152b37c35
OpenVINO Stack

Internet of Things Group

Layered access: every user can choose the proper API level for integration.

Surveillance App
Retail App
Automotive App
Industrial App

OpenVINO Toolkit
Reference Algorithms
Open Model Zoo
Tools: train, optimize, profile
OpenCV
DL Inference Engine
Optimized Kernel Libraries
CPU
GPU
VPU
FPGA

Products
Algorithms
Models
APIs
Kernels
OpenVINO Stack

Layered access: every user can choose the proper API level for integration.

- Surveillance App
- Retail App
- Automotive App
- Industrial App

OpenVINO Toolkit
- Reference Algorithms
- Open Model Zoo
- Tools: train, optimize, profile
- OpenCV
- DL Inference Engine
- Optimized Kernel Libraries
  - CPU
  - GPU
  - VPU
  - FPGA
  - ...

Products
- Super Users
- Advanced Users
- Regular Users
- Beginners

Algorithms
- Kernels
- APIs
- Models

Internet of Things Group
Open Model Zoo is a Software 2.0 library!
<table>
<thead>
<tr>
<th></th>
<th>OpenCV (SW 1.0)</th>
<th>Open Model Zoo (SW 2.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Computational unit</strong></td>
<td>Function</td>
<td>Model</td>
</tr>
</tbody>
</table>
| **Development tools** | Compiler, Debugger, Profiler, Build System, etc. | • Training Frameworks (PyTorch, TensorFlow, Caffe, etc.)  
• Deployment Toolkits (OpenVINO, TVM, OpenCV, etc.)  
• Annotation tools (CVAT, Vatic, etc.) |
| **Binary distribution** | Operating system shared object (.dll/.so) | Intermediate Representation (.onnx, .pb.frozen, .prototxt/.caffemodel, .xml/.bin, etc.) |
| **Source distribution** | Source code is available (e.g. via GitHub) | Checkpoint + Training Code (+ Dataset) is available |
Current stat(u)s – OpenVINO R1’2019

- Inference Models: 41 public + 37 intel
  - **Areas:** Object Detection, Object Recognition, Reidentification, 2D/3D Semantic Segmentation, Pose estimation, Image Processing, Text Detection
  - **Precision:** FP32, FP16 + INT8, INT1 (for selected models)
- 12 Intel models have training source
  - Vehicle/License plate detection and recognition, Person detection, Human Pose estimation, Face Detection and Recognition, Instance Segmentation, Super Resolution, Action recognition
- Demos: 15 C++ and 5 Python

https://github.com/opencv/openvino_training_extensions
Edge networks design rationale
Algo optimization: not all networks are created equal

Algo optimization

• There's a substantial amount of evidence that widespread ("default") network architectures can be "compressed" during training for applied problems

• 2x-10x speed ups w/o significant loss in accuracy are fairly common

<table>
<thead>
<tr>
<th>Network</th>
<th>mAP</th>
<th>Params</th>
<th>MAdd</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD300[34]</td>
<td>23.2</td>
<td>36.1M</td>
<td>35.2B</td>
<td>-</td>
</tr>
<tr>
<td>SSD512[34]</td>
<td>26.8</td>
<td>36.1M</td>
<td>99.5B</td>
<td>-</td>
</tr>
<tr>
<td>YOLOv2[35]</td>
<td>21.6</td>
<td>50.7M</td>
<td>17.5B</td>
<td>-</td>
</tr>
<tr>
<td>MNet V1 + SSDLite</td>
<td>22.2</td>
<td>5.1M</td>
<td>1.3B</td>
<td>270ms</td>
</tr>
<tr>
<td>MNet V2 + SSDLite</td>
<td>22.1</td>
<td>4.3M</td>
<td>0.8B</td>
<td>200ms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network</th>
<th>Top 1</th>
<th>Params</th>
<th>MAdd</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNetV1</td>
<td>70.6</td>
<td>4.2M</td>
<td>575M</td>
<td>113ms</td>
</tr>
<tr>
<td>ShuffleNet (1.5)</td>
<td>71.5</td>
<td>3.4M</td>
<td>292M</td>
<td>-</td>
</tr>
<tr>
<td>ShuffleNet (x2)</td>
<td>73.7</td>
<td>5.4M</td>
<td>524M</td>
<td>-</td>
</tr>
<tr>
<td>NasNet-A</td>
<td>74.0</td>
<td>5.3M</td>
<td>564M</td>
<td>183ms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network</th>
<th>Top 1</th>
<th>Params</th>
<th>MAdd</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNetV2</td>
<td>72.0</td>
<td>3.4M</td>
<td>300M</td>
<td>75ms</td>
</tr>
<tr>
<td>MobileNetV2 (1.4)</td>
<td>74.7</td>
<td>6.9M</td>
<td>585M</td>
<td>143ms</td>
</tr>
</tbody>
</table>


MS COCO (Object Detection)

MS COCO (Human Pose)

• Find the sweet spot (80/20 rule) on a Pareto curve for a practical and pragmatic implementation

• Use application-specific datasets instead of (in addition to) public datasets
Training on problem-specific data
# Academic vs Production Datasets

## COCO Explorer

COCO 2017 train/val browser (123,287 images, 886,284 instances)

<table>
<thead>
<tr>
<th>Type of object</th>
<th>Number of unique objects</th>
<th>Number of Frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pedestrian</td>
<td>1,020,114</td>
<td>7,997,092</td>
</tr>
<tr>
<td>Vehicle</td>
<td>709,843</td>
<td>7,282,679</td>
</tr>
<tr>
<td>Ignored</td>
<td>258,397</td>
<td>6,941,283</td>
</tr>
<tr>
<td>Non-vehicle</td>
<td>127,125</td>
<td>7,008,948</td>
</tr>
</tbody>
</table>

- ~65x bigger than MS COCO in size
- ~$250K worth of data and annotation (excl. R&D costs)
- Contains data specifically for a surveillance scenario
R&D Lifecycle in Numbers

Collect/Purchase dataset for training
• $0.01-30 per image
• $1-100 per hour
• $0.03-1 per annotated object

Build training infrastructure
• 10-100 computational nodes
• 100-1000 GBs of storage and network transfers

Train the model
• 1-10 Data Scientists
• 3-12 months

Deploy & Monitor

Improve training
Add more data
OpenVINO Training Extensions

• Subset of models from Open Model Zoo is available in their original format at https://github.com/opencv/openvino_training_extensions

• Every model has a respective folder with:
  • **Model** in the original format of the training framework
  • **Scripts to train or fine-tune** the model on **sample dataset**
  • **Documentation** on how to run and use the scripts
Key takeaways

• Deep Learning requires us to rethink the tooling we have for software
• Open Model Zoo is a collection of algorithms for POCs or production
• Algorithm optimization is highly effective in Deep Learning
• Data/Training infrastructure is a major cost in development process
Links

https://github.com/opencv/open_model_zoo

https://github.com/opencv/openvino_training_extensions

https://github.com/opencv/cvat
OMZ: Suggested Workflow for R&D

1. Find a “similar” demo/sample app
2. Try various models (public or intel)
3. Build a POC using models with best perf/accuracy

- Fine-tune on your data
- Use compression techniques
- Switch HW
## Software 1.0 vs 2.0

<table>
<thead>
<tr>
<th></th>
<th>Software 1.0</th>
<th>Software 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Programmed by</strong></td>
<td>Writing code in C++/Java/Python/etc.</td>
<td>Accumulating / annotating / cleaning datasets</td>
</tr>
<tr>
<td><strong>Fixing bugs</strong></td>
<td>Find the “line” in source code and change it</td>
<td>Find the corner case and augment training procedure with more data</td>
</tr>
<tr>
<td><strong>Pros</strong></td>
<td>• Relatively easy to interpret&lt;br&gt;• Mature ecosystem</td>
<td>• Superhuman in a range of verticals&lt;br&gt;• Computationally homogeneous&lt;br&gt;• Very agile</td>
</tr>
</tbody>
</table>
Hype Cycle

https://en.wikipedia.org/wiki/Hype_cycle
Open Model Zoo: Architecture

Demos

Models

Model Downloader

Public Models
Best known method to communicate on platform capabilities between teams

Intel Models
- Wide range of common tasks
- Solid perf/accuracy trade-offs

Advanced reference use cases

Large collection of models running with OpenVINO