Deep Neural Network Design Automation

Bei Yu

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Computer Vision

- Humans use their eyes and their brains to visually sense the world.
- Computers use their cameras and computation to visually sense the world.

Jian Sun, “Introduction to Computer Vision and Deep Learning”. 
Few More Core Problems

Classification

Detection

Segmentation

Sequence

Image → Region → Pixel → Video
Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)

11x11 conv, 96, /4, pool/2

5x5 conv, 256, pool/2

3x3 conv, 384

3x3 conv, 384

3x3 conv, 256, pool/2

fc, 4096

fc, 4096

fc, 1000
Revolution of Depth

**AlexNet, 8 layers**
* (ILSVRC 2012)

**VGG, 19 layers**
* (ILSVRC 2014)

**GoogleNet, 22 layers**
* (ILSVRC 2014)

Slide Credit: He et al. (MSRA)
Revolution of Depth

AlexNet, 8 layers (ILSVRC 2012)

VGG, 19 layers (ILSVRC 2014)

ResNet, 152 layers (ILSVRC 2015)
Some Recent Classification Architectures

- **AlexNet** (Krizhevsky, Sutskever, and E. Hinton 2012) 233MB
- **Network in Network** (Lin, Chen, and Yan 2013) 29MB
- **VGG** (Simonyan and Zisserman 2015) 549MB
- **GoogleNet** (Szegedy, Liu, et al. 2015) 51MB
- **ResNet** (K. He et al. 2016) 215MB
- **Inception-ResNet** (Szegedy, Vanhoucke, et al. 2016)
- **DenseNet** (Huang et al. 2017)
- **Xception** (Chollet 2017)
- **MobileNetV2** (Sandler et al. 2018)
- **ShuffleNet** (Zhang, Zhou, et al. 2018)
Some Recent Classification Architectures

- **AlexNet** (Krizhevsky, Sutskever, and E. Hinton 2012) 233MB
- **Network in Network** (Lin, Chen, and Yan 2013) 29MB
- **VGG** (Simonyan and Zisserman 2015) 549MB
- **GoogleNet** (Szegedy, Liu, et al. 2015) 51MB
- **ResNet** (K. He et al. 2016) 215MB
- **Inception-ResNet** (Szegedy, Vanhoucke, et al. 2016) 23MB
- **DenseNet** (Huang et al. 2017) 80MB
- **Xception** (Chollet 2017) 22MB
- **MobileNetV2** (Sandler et al. 2018) 14MB
- **ShuffleNet** (Zhang, Zhou, et al. 2018) 22MB
1

Fei-Fei Li & Justin Johnson & Serena Yeung, Stanford cs231n.
When Machine Learning Meets Hardware

Convolution layer is one of the most expensive layers

▶ Computation pattern
▶ Emerging challenges

More and more end-point devices with limited memory

▶ Cameras
▶ Smartphone
▶ Autonomous driving
1st Challenge: Model Size

Hard to distribute large models through over-the-air update

2nd Challenge: Energy Efficiency

AlphaGo: 1920 CPUs and 280 GPUs, $3000 electric bill per game

on mobile: drains battery
on data-center: increases TCO

---

Outline

Algorithmic Level

Architecture Level

Compilation Level

Hardware Implementation Level

Physical Synthesis Level
Outline

Algorithmic Level

Architecture Level

Compilation Level

Hardware Implementation Level

Physical Synthesis Level
Dive Deeper Into Box for Object Detection

Ran Chen¹, Yong Liu², Mengdan Zhang², Shu Liu³, Bei Yu¹, and Yu-Wing Tai²

¹ The Chinese University of Hong Kong
{rchen,byu}@cse.cuhk.edu.hk
² Tencent Youtu Lab
{ly.chaos,zhangmengdanz,yuwingtai}@gmail.com
³ SmartMore
sliu@smartmore.com
Algorithm EX2: Semantic Segmentation [ECCV’20]

Tensor Low-Rank Reconstruction for Semantic Segmentation

Wanli Chen\textsuperscript{1}, Xinge Zhu\textsuperscript{1}, Ruqi Su\textsuperscript{2}, Junjun He\textsuperscript{2}, Ruiyu Li\textsuperscript{3}, Xiaoyong Shen\textsuperscript{3}, and Bei Yu\textsuperscript{1}

\textsuperscript{1} The Chinese University of Hong Kong
\{wlchen,byu\}@cse.cuhk.edu.hk, zx018@ie.cuhk.edu.hk
\textsuperscript{2} Shanghai Jiao Tong University
\{hejunjun,ruoquisun\}@sjtu.edu.cn
\textsuperscript{3} SmartMore
\{ryli,xiaoyong\}@smartmore.com

\begin{itemize}
\item[(a)] Input Image
\item[(b)] Tensor Generation Module (TGM)
\item[(c)] Tensor Reconstruction Module (TRM)
\item[(d)] Final Prediction
\end{itemize}
Outline

Algorithmic Level

Architecture Level

Compilation Level

Hardware Implementation Level

Physical Synthesis Level
Neural Architecture Search (NAS)

- Designing neural architecture is extremely challenging.

- Mechanism of neural networks is not well interpreted.

- Can we advance AI/ML using artificial intelligence instead of human intelligence?
Neural Architecture Search (NAS)

- **Search space**
- **Sample networks** \( \pi_\theta \)
- **Update policy**
- **Environment as a black box**
- **Training and evaluation**
- **Hardware simulation**
Neural Architecture Search (NAS)

Black box Optimization

- Find the optimal network configuration to maximize the performance.
- Huge search space: e.g. $1.28 \times 10^{54}$ settings.

Available methods

- Reinforcement learning.
- Evolutionary algorithm.
- Differentiable architecture search.
Im2col (Image2Column) Convolution

Transform convolution to matrix multiplication

Unified calculation for both convolution and fully-connected layers
Compression Approach: Sparsity$^1,^2$

\[ \mathbf{X} \in \mathbb{R}^{d \times (k^2 c)} \times \mathbf{S} \in \mathbb{R}^{(k^2 c) \times n} = \mathbf{Y} \in \mathbb{R}^{d \times n} \]

Sparse DNN

- **Sparsification**: weight pruning;
- **Compression**: compressed sparse format for storage;
- **Potential acceleration**: sparse matrix multiplication algorithm.

---


$^2$Yihui He, Xiangyu Zhang, and Jian Sun (2017). “Channel Pruning for Accelerating Very Deep Neural Networks”. In: Proc. ICCV.
Compression Approach: Low-Rank$^1,^2$

Low-rank DNN

- **Low-rank approximation**: matrix decomposition or tensor decomposition.
- **Compression and acceleration**: less storage required and less FLOP in computation.

---


Non-linearity Approximation

Activation unit: ReLU

Error more sensitive to positive response;

Enlarge the solution space.

\[
\min_W \sum_{i=1}^{N} \|WX_i - Y_i\|_F \rightarrow \min_W \sum_{i=1}^{N} \|r(WX_i) - Y_i\|_F
\]

- \(X\): input feature map
- \(Y\): output feature map

Our Idea: Unified Structure\(^1\) (Best Student Paper Award)

Simultaneous low-rank approximation and network sparsification;
Non-linearity is taken into account.
Acceleration is achieved with structured sparsity.

\(^1\)Yuzhe Ma et al. (2019). “A Unified Approximation Framework for Non-Linear Deep Neural Networks”. In: Proc. ICTAI.
Outline

Algorithmic Level

Architecture Level

Compilation Level

Hardware Implementation Level

Physical Synthesis Level
Deep Learning Compiler - TVM

Global View of TVM Stack

Frameworks
- Caffe2
- CNTK
- CoreML
- Pytorch (caffe2, cntk supported via onnx)

Computational Graph

Graph Optimizations

Tensor Expression Language

Schedule Primitives Optimization

Accelerators
- CUDA
- ARM
- Vulkan
- AMDGPUs
- OpenCL
- Javascript/WASM
- X86

TVM: End to End Optimization

Computational Graph Optimization: Operator Fusion

- conv2d
- bn
- relu
- fused-conv2d-bn-relu

Frameworks: open, ML, etc.

Hardware: various devices and platforms
TVM: End to End Optimization

Layer-wise Optimization: Autotuning

Tuning algorithms:
- Active learning.
- Transfer learning.
- Reinforcement learning.
TVM Domain Specific Language

Decoupling scheduling and algorithms.

▸ Specify the algorithm.

▸ Specify the schedule.
TVM/VTA: Full Stack Open Source System

- JIT compile accelerator micro code.
- Support heterogenous devices, 10x better than CPU on the same board.
- Move hardware complexity to software.
TVM: End-to-End Integration
TVM Domain Specific Language + Loop Tiling

- Optimize data locality
- Minimize memory conflict
- Optimize for device cache
TVM Domain Specific Language + New Features

- Allow read and write to special memory scope
- Allow hook into hardware instructions
- Allow optimize for pipeline parallelism via reordering

```python
+ Cache Data on Accelerator Special Buffer
CL = s.cache_write(C, vdla.acc_buffer)
AL = s.cache_read(A, vdla.inp_buffer)
# additional schedule steps omitted ...

+ Map to Accelerator Tensor Instructions
s[CL].tensorize(yi, vdla.gemm8x8)

inp_buffer AL[8][8], BL[8][8]
acc_buffer CL[8][8]
for yo in range(128):
    for xo in range(128):
        vdla.fill_zero(CL)
        for ko in range(128):
            vdla.dma_copy2d(AL, A[ko*8:ko*8+8][yo*8:yo*8+8])
            vdla.dma_copy2d(BL, B[ko*8:ko*8+8][xo*8:xo*8+8])
            vdla.fused_gemm8x8_add(CL, AL, BL)
        vdla.dma_copy2d(C[yo*8:yo*8+8,xo*8:xo*8+8], CL)
```
TVM: End to End Optimization

[Diagram showing computational graph optimization and hardware]

Frameworks

Computational Graph Optimization

Hardware
TVM: Blackbox Autotuning

High experiment cost
TVM: Statistical Cost Model based Approach

Learn from historical data
Outline

Algorithmic Level

Architecture Level

Compilation Level

Hardware Implementation Level

Physical Synthesis Level
Object Detection on FPGA

- [http://www.pynq.io/](http://www.pynq.io/)
- DAC-2018 System Design Contest

Sponsored by:

- [DJI](https://www.dji.com/)
- [NVIDIA](https://www.nvidia.com/)
- [Xilinx](https://www.xilinx.com/)
Overall System Diagram

Input Image 360*640
Resize to 160*320
Convert to RGB

Copy to DRAM

Image RGB data

feature map after each pooling layer

model weights

on-chip data transfer

Coordinates

off-chip data transfer

PS

PL

BRAM

Image Normalization

Depth-Wise Conv 3x3

Point-Wise Conv 1x1

Max Pooling

Bounding Box Regression

weight buffer 3x3

weight buffer 1x1

buffer 1
buffer 2
buffer 3
buffer 4
buffer 5
buffer 6
... buffer 14
buffer 15
buffer 16
Image partition, fine grained buffer scheduling, IP pipeline

Image chunk

Feature map chunks

Feature map chunks

DRAM data transfer pattern

Fine Grained IP/Buffer Scheduling

Input image: cut into 8x8 slices

DRAM

BRAM

DRAM

DRAM data transfer

Layer 1: 3x3 CONV IP

Layer 2: 1x1 CONV IP

Layer 3: POOLING IP

DDR load

DDR write

1/3 Channels

2/3 Channels

3/3 Channels

3x3 CONV IP

1x1 CONV IP

POOLING IP

DDR load

DDR write

b1

b2

b3

b4

b5

b6

b5

b5

b5

b6

b6

b6

 DDR write 1/3 Channels

 DDR write 2/3 Channels

 DDR write 3/3 Channels
FPGA Design Flow

1. DNN Design in C/C++ *(example)*
2. Generate RTL *(example)* by tool *Vivado HLS*
3. Generate bitstream by tool *Vivado IDE*
4. Load bitstream to FPGA board
Outline

Algorithmic Level

Architecture Level

Compilation Level

Hardware Implementation Level

Physical Synthesis Level
Huawei Ascend 910

Adder is one of the most important component!

(a) Logic perspective

(b) Physical synthesis perspective
EDA Challenges: How to Design an AI Chip Component?

(c) Current EDA tool output

(d) Manual design

C/C++ Programming skills are heavily required – welcome students with ICPC background