Model Compression for Efficient AI Computing

From TinyML to LargeML

Song Han
MIT
songhan.mit.edu
tinyml.mit.edu
@SongHan/MIT
Model Compression
Bridges the Gap between the Supply and Demand of Computation

- Specialized hardware is important but not enough.
- Model compression bridges the gap.
Model Compression

Applications

Large Language Model

Generative AI

Advanced Driver Assistance System

TinyML

Techniques

Hardware-aware NAS

Pruning & Sparsity

Quantization

Distillation

New Primitive

A language model is a type of artificial intelligence that can process and generate human-like language, based on vast amounts of data it has been trained on.
Efficient Large Language Models
Reducing LLM Serving Cost and Accelerating Inference

ChatGPT is at capacity right now

Get notified when we're back

Write a rap about the status of ChatGPT.

Yeah, yo, what's up ChatGPT fam
A lot of people trying to jam
But don't worry, we got your back
Just check back soon, we'll get on track

ChatGPT's the place to be
For all your AI chat needs
We're working hard to keep up the pace
So hold tight, we'll be back in this space

So don't you worry, don't be mad
We're working hard, it's not so bad
Just give us time, we'll be back
ChatGPT, the future of chat, that's a fact.

The LLM serving costs are extremely high

Due to high demand, we've temporarily paused upgrades.
SpAtten: Transformer with Sparse Attention

Token Pruning: not every token are created equal

As a visual treat, the film is almost perfect.
11 Tokens ↓ 8 Heads

BERT Layer 1 (100% Computation & Memory Access)

As treat, film perfect.
6 Tokens ↓ 5 Heads

Layer 2 (34%)

film perfect
2 Tokens ↓ 4 Heads

Layer 3 (9%)

Sentiment Classification: Positive ✓

Remove redundant token and head according to cumulative importance

SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning [HPCA'21]
SpAtten: Transformer with Sparse Attention

Progressive Quantization: high confident attention requires low bit width

Specialized top-k engine to select important token and head

Speed Over TITAN Xp GPU

22.1× speedup with specialized datapath (ASIC)
3.4× speedup with cascade token & head pruning
2.8× speedup with progressive quantization
Lite Transformer
Local Convolution + Global Attention

- **Long-Short Range Attention (LSRA):**
  - **Convolution:** Efficiently extract the **local** (short-range) features.
  - **Attention:** Tailored for **global** (long-range) feature extraction.

**Original Attention**
(Too much emphasize on local feature extraction)

**Attention in LSRA**
(Dedicated for global feature extraction)

---

Lite Transformer with Long-Short Range Attention [ICLR 2020]
Same Principle, Diverse Applications

Applications

- Large Language Model
- Generative AI
- Advanced Driver Assistance System
- TinyML

Techniques

- Hardware-aware NAS
- Pruning & Sparsity
- Quantization
- Distillation
- New Primitive

A large language model is a type of artificial intelligence that can process and generate human-like language, based on vast amounts of data it has been trained on.
Compressing and Accelerating Diffusion Models

Prompt: A fantasy landscape, trending on artstation.

Original

Edited

Stable Diffusion+SDEdit:
1855GMACs, 369ms

Ours:
225GMACs (8.2x),
51.2ms (7.2x)

Efficient Spatially Sparse Inference for Conditional GANs and Diffusion Models, NeurIPS’22
Spatially Sparse Inference for Diffusion Models

Vanilla Model Wastes Many Computations to Re-synthesize the Entire Image

- Only 1.7% region is edited, but vanilla model re-synthesizes the entire image.
- Feature maps remain mostly the same at unedited regions.
- Reuse cached activations to selectively update edited regions (8× less computation).

Efficient Spatially Sparse Inference for Conditional GANs and Diffusion Models, NeurIPS’22
Spatially Sparse Inference for Diffusion Models

Efficiency Results of SIGE

<table>
<thead>
<tr>
<th>Latency (ms)</th>
<th>DDIM</th>
<th>SIGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency on NVIDIA RTX 3090</td>
<td>37.5×</td>
<td>12.6×</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Latency (s)</th>
<th>Latency on Apple M1 Pro CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency on NVIDIA RTX 3090</td>
<td>12.9×</td>
</tr>
</tbody>
</table>

Efficient Spatially Sparse Inference for Conditional GANs and Diffusion Models, NeurIPS’22
Spatially Sparse Inference for Diffusion Models

Qualitative Results of SIGE on Stable Diffusion

A photograph of a horse on a grassland.

Original
11.6% Masked

Stable Diffusion: 1855GMACs 369ms

Ours: 514G (3.6x) 95.0ms (3.9x)

Image Inpainting

Latency Measured on NVIDIA RTX 3090

A fantasy beach landscape, trending on artstation.

Original
2.9% Edited

Stable Diffusion+SDEdit: 1855GMACs 369ms

Ours: 353G (5.3x) 76.4ms (4.8x)

Image Editing

Efficient Spatially Sparse Inference for Conditional GANs and Diffusion Models, NeurIPS’22
Same Principle, Diverse Applications

Applications

Large Language Model

Generative AI

Advanced Driver Assistance System

TinyML

Techniques

Hardware-aware NAS

Pruning & Sparsity

Quantization

Distillation

New Primitive
PVCNN: Point-Voxel CNN
New primitive for handling spatial sparsity in point clouds

Voxel-Based Models
- 3D ShapeNets [CVPR'15]
- VoxNet [IROS'15]
- 3D U-Net [MICCAI'16]

Point-Based Models
- PointNet [CVPR'17]
- PointCNN [NeurIPS'18]
- DGCNN [SIGGRAPH'19]

Voxel Resolution
- 128 x 128 x 128 resolution
- 83 GB (Titan XP x 7)
- 7% information loss
- 64 x 64 x 64 resolution
- 11 GB (Titan XP x 1)
- 42% information loss

GPU memory consumption increases cubically with the volumetric resolution

Point-based models suffer from random memory accesses and dynamic kernel computation
**SPVNAS accelerated by TorchSparse**

TorchSparse brings about another 1.3x speedup to the efficient SPVNAS model

Measured on GTX1080Ti

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean IoU</th>
<th>Throughput</th>
<th>Parameters</th>
<th>FLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinkowskiNet</td>
<td>63.1</td>
<td>3.4 FPS</td>
<td>21.7M</td>
<td>114.0G</td>
</tr>
<tr>
<td>SPVNAS (Ours)</td>
<td>63.6</td>
<td>9.1 FPS</td>
<td>2.6M</td>
<td>15.0G</td>
</tr>
<tr>
<td>SPVNAS + TorchSparse (Ours)</td>
<td>63.6</td>
<td>12.1 FPS</td>
<td>2.6M</td>
<td>15.0G</td>
</tr>
</tbody>
</table>

System-Algorithm Codesign

Efficient 3D Network

Sparse Inference Engine

SPVNAS

SPVNAS + TorchSparse
TorchSparse: Efficient Point Cloud Library

Balancing computation regularity and computation overhead

- Automatically determine the best tradeoff between \textit{control flow} and \textit{computation} overhead;
- \textit{Mixed dataflow configurations} for different layers and forward/backward computation.

Implicit GEMM without zero-skipping

Zero-skipping Implicit GEMM (weight split=1)

Zero-skipping Implicit GEMM (weight split=3)

Data Driven Auto-tuning based on profiling feedback
PointAcc: Point Cloud Accelerator

One hardware architecture, diverse neural architectures

- The sparsity of point clouds leads to two bottlenecks:
  - **Diverse mapping operations** for searching the input, output, weight maps (e.g., k-Nearest Neighbor, Ball Query, Kernel Mapping)
  - Data movement overhead from **gather and scatter** of the sparse features
- PointAcc maps diverse mapping ops into sort-based computation with **one versatile hardware architecture**.
- PointAcc reduces off-chip memory access and minimize the overhead of gather and scatter by **flexible caching** and **layer fusion**.

Diverse Mapping Ops in One Versatile Architecture

![Diagram of diverse mapping operations](image)

### Without Layer Fusion

<table>
<thead>
<tr>
<th>DRAM access per point</th>
<th>FC</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>write</td>
</tr>
</tbody>
</table>

| Without layer fusion | With layer fusion |

### With Layer Fusion

<table>
<thead>
<tr>
<th>DRAM access per point</th>
<th>FC</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>write</td>
</tr>
</tbody>
</table>

PointAcc: Efficient Point Cloud Accelerator [MICRO-54, 2021]
BEVFusion: Dense (Camera) + Sparse (LiDAR)

Accelerate LSS by 40x with Interval Reduction and Pre-computation

- Ranked first on nuScenes 3D object detection (2022/6).
- Ranked first on nuScenes 3D object tracking (2022/7).
- Ranked first on Waymo 3D object detection (2022/11).
- Ranked first on Argoverse 3D object detection (2023/4).
PhD Student — Zhijian Liu
Research Interest: **Efficient Algorithms and Systems for Deep Learning**
Graduating in Fall’23

Zhijian Liu
zhijian@mit.edu
https://zhijianliu.com
@zhijianliu96

Representative Work
- **Algorithms:**
  - PVCNN (NeurIPS’19 Spotlight)
  - SPVNAS (ECCV’20, TPAMI’21)
  - FlatFormer (CVPR’23)
- **Systems:**
  - TorchSparse (MLSys’22)
- **Applications:**
  - BEVFusion (ICRA’23)

Selected Honors
- Qualcomm Innovation Fellowship
- NVIDIA Graduate Fellowship
- MIT Ho-Ching and Han-Ching Fund Award
Same Principle, Diverse Applications

Applications

- Large Language Model
- Generative AI
- Advanced Driver Assistance System
- TinyML

Techniques

- Hardware-aware NAS
- Pruning & Sparsity
- Quantization
- Distillation
- New Primitive
Background: The Era of AIoT on Microcontrollers

- **Problem**: restricted memory size

<table>
<thead>
<tr>
<th></th>
<th>Cloud AI</th>
<th>Mobile AI</th>
<th>Tiny AI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory (Activation)</td>
<td>32GB</td>
<td>4GB</td>
<td>320kB</td>
</tr>
<tr>
<td>Storage (Weights)</td>
<td>~TB/PB</td>
<td>256GB</td>
<td>1MB</td>
</tr>
</tbody>
</table>
MCUNet
Deploy AI on MCUs that has only 256KB SRAM

Face/mask detection

Person detection

The camera is OpenMV Cam.
Inference Is Good. Can We Learn on Edge?

AI systems need to continually adapt to new data collected from the sensors. Not only inference, but also training.

- On-device learning: **better privacy, lower cost, customization, life-long learning**
- Training is more **expensive** than inference, hard to fit edge hardware (limited memory)

The camera is OpenMV Cam.
Sparse Training

Only update important layers and sub-tensors to save memory

- **Sensitivity analysis**
  - *Later layers* are more important
  - The first point-wise conv in each block contributes more

- **Detailed Update Scheme for MobileNetV2**
  - The activation cost is high for the early layers;
  - The weight cost is high for the later layers;
  - The overall memory cost is low for the middle layers.
    - Bias-only update
    - Update weights for the middle layers

On-Device Training Under 256KB Memory [Lin et al., NeurIPS 2022]
Low-Precision Training
with Quantization Aware Scaling (QAS)

- Optimizing an INT8 quantized graph leads to memory and computing savings
  - All weights and activations are in INT8
  - Different from quantization-aware training (QAT), where operations are performed in FP16

- … But at the cost of worse convergence
  - We found the issue lies in gradient scale mismatch

- Solution: quantization-aware scaling (QAS)

\[
W = s_W \cdot (W/s_W)_{\text{quantize}} \approx s_W \cdot W_Q, \quad G_{W_Q} \approx s_W \cdot G_W
\]

weight and gradient ratios are off by \( s_W^{-2} \)

\[
\|W_Q\|/\|G_{W_Q}\| \approx \|W/s_W\|/\|s_W \cdot G_W\| = s_W^{-2} \cdot \|W\|/\|G\|
\]

Thus, we need to re-scale the gradients \( G'_{W_Q} = G_{W_Q} \cdot s_W^{-2} \)

QAS improves the val performance.

![Graph showing improvements in top-1 accuracy and val loss with and without QAS]
Tiny Training Engine

Translate the theoretical saving into measured savings. Runtime => Compile time

Python Defined Models

Traced Static Graph

Backward Graph

Compile-time AutoDiff

Forward Graph

Graph Opt.

IR

Tune Schedules

CodeGen

Executable Binaries for Training

---

6x speedup compared to TensorFlow on Jetson Nano
**Device:** Jetson Nano; **Backend:** Tiny Training Engine; **Task:** Speech Recognition

### TTE On-Device Learning of Wave2Vec

<table>
<thead>
<tr>
<th>bs=1</th>
<th>bs=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency (ms)</td>
<td></td>
</tr>
<tr>
<td>2,337</td>
<td>547</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bs=1</th>
<th>bs=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory (MB)</td>
<td></td>
</tr>
<tr>
<td>2,971</td>
<td>843</td>
</tr>
</tbody>
</table>

### Word Error Rate on TIMIT

<table>
<thead>
<tr>
<th>Wav2Vec2.0</th>
<th>Word Error Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full update</td>
<td>33.9</td>
</tr>
<tr>
<td>Classifier-only update</td>
<td>98.7</td>
</tr>
<tr>
<td>Sparse Update (last two Encoder + FC)</td>
<td>33.3</td>
</tr>
</tbody>
</table>
Model Compression for Diverse Applications

- Predictive Maintenance
- Gesture Recognition
- Video Recognition
- Health Monitoring
- Art Generation
- Video Synthesis
- Music Composition
- Fashion Design
- Question Answering
- Storytelling
- Search Engine Revolution
- Augmented Reality
- Autonomous Driving
- Sentiment Analysis
- Blind Spot Detection
- Machine Translation
- Adaptive Cruise Control
- Chatbots
- Video Synthesis
- Gesture Recognition
- Health Monitoring
- Large Language Model
- Generative AI
- Driver Assistance System
- TinyML

Application (demand of computation)

Hardware (supply of computation)
MCUNet: Tiny Deep Learning on IoT Devices [Lin et al., NeurIPS 2020]
MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning [Lin et al., NeurIPS 2021]
On-Device Training Under 256KB Memory [Lin et al., NeurIPS 2022]
Open Source

MCUNet: Tiny Deep Learning for Microcontrollers

This is the official implementation of TinyEngine, a TinyEngine is a part of MCUNet, a co-design framework for tiny deep learning on microcontrollers. TinyEngine is a part of MCUNet, a co-design framework for tiny deep learning on microcontrollers.

The MCUNet and TinyNAS repo is here.

MCUNetV1 | MCUNetV2 | MCUNetV3

Sign up here to get updates!

https://forms.gle/UW1uUmnfk1k6UJPPA
New Course: TinyML and Efficient Deep Learning Computing


TinyML and Efficient Deep Learning
6.S965 • Fall 2022 • MIT

Have you found it difficult to deploy neural networks on mobile devices and IoT devices? Have you ever found it too slow to train neural networks? This course is a deep dive into efficient machine learning techniques that enable powerful deep learning applications on resource-constrained devices. Topics cover efficient inference techniques, including model compression, pruning, quantization, neural architecture search, and distillation; and efficient training techniques, including gradient compression and on-device transfer learning, followed by application-specific model optimization techniques for videos, point cloud, and NLP; and efficient quantum machine learning. Students will get hands-on experience implementing deep learning applications on microcontrollers, mobile phones, and quantum machines with an open-ended design project related to mobile AI.

- **Time:** Tuesday/Thursday 3:30-5:00 pm Eastern Time
- **Location:** 36-156
- **Office Hour:** Thursday 5:00-6:00 pm Eastern Time, 36-344 Meeting Room
- **Discussion:** Piazza
- **Homework submission:** Canvas
- **Online lectures:** The lectures will be streamed on YouTube.
- **Resources:** MIT HAN Lab, Github, TinyML, MCI/Net, GFA
- **Contact:** Students should ask all course-related questions on Piazza. For external inquiries, personal matters, or emergencies, you can email us at 6s965-fa2022-staff@mit.edu.

Instructor Song Han
Email: songhan@mit.edu

TA Zhiqian Liu
Email: zhiqian@mit.edu

TA Yujun Lin
Email: yujunlin@mit.edu

Anonymous Student Feedback Collected from Mid-term

- This course is a deep dive into efficient machine learning techniques that enable powerful deep learning applications on resource-constrained devices.

I really like how structured the labs are, and being able to see actual implementations of the techniques we learn about.

This is honestly one of the best set up courses I’ve taken at MIT

I love how we are using microcontroller and focusing on application instead of just theories.

I managed the weekly labs and lectures by only watching the course on YouTube. As a researcher, I gained some valuable knowledge from your course. Excellent slides and teaching and useful labs.

I like the class and I have been able to follow the class easily (which had rarely happened to me in my previous courses)
MIT AI Hardware Program

MIT Microsystems Technology Laboratories (SoE)
MIT Quest for Intelligence – Corporate (SCC)

Co-Leads: Jesús del Alamo and Aude Oliva
Internal Advisory Board Chair: Anantha Chandrakasan
TinyML and Efficient AI

Sponsors:

Media: