

TinyML and Efficient Deep Learning Computing



Song Han

Associate Professor, MIT
Distinguished Scientist, NVIDIA
<https://efficientml.ai>

 @SongHan/MIT

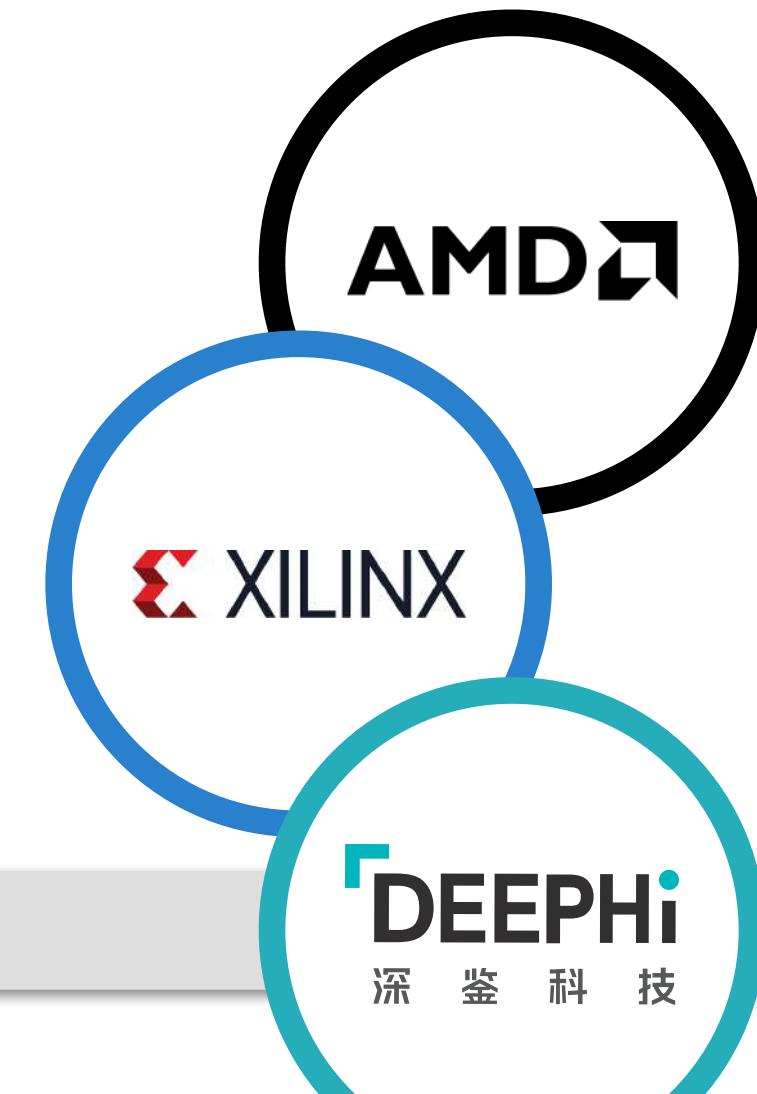


Song Han

- Ph.D. from Stanford University, advised by Prof. Bill Dally
- Deep Compression (best paper award of ICLR)
- EIE (top5 cited paper in 50 years of ISCA)

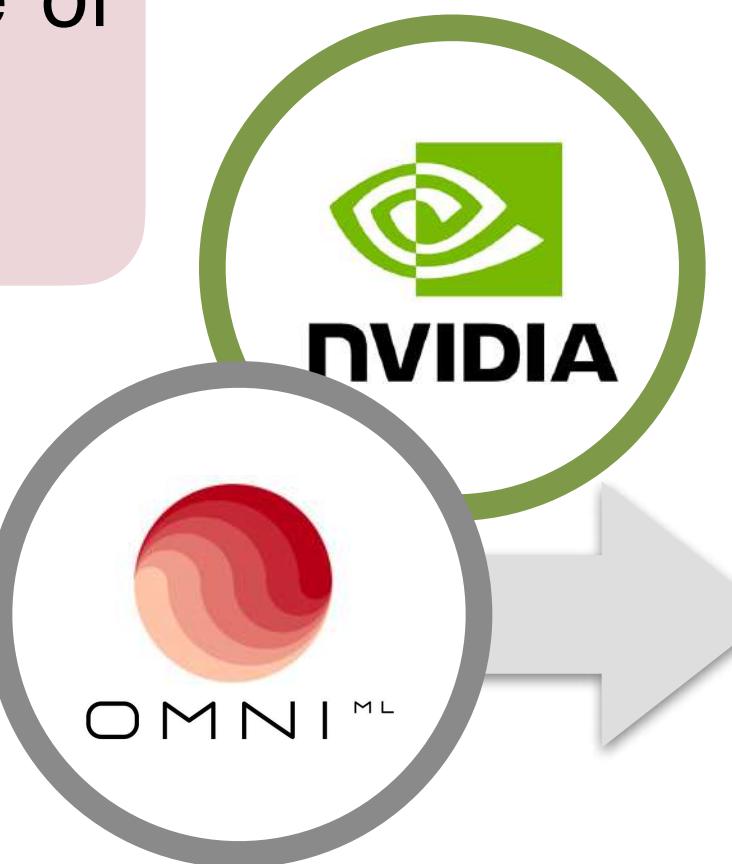


- B.S. from Tsinghua University



- Cofounder of DeePhi (now part of AMD)

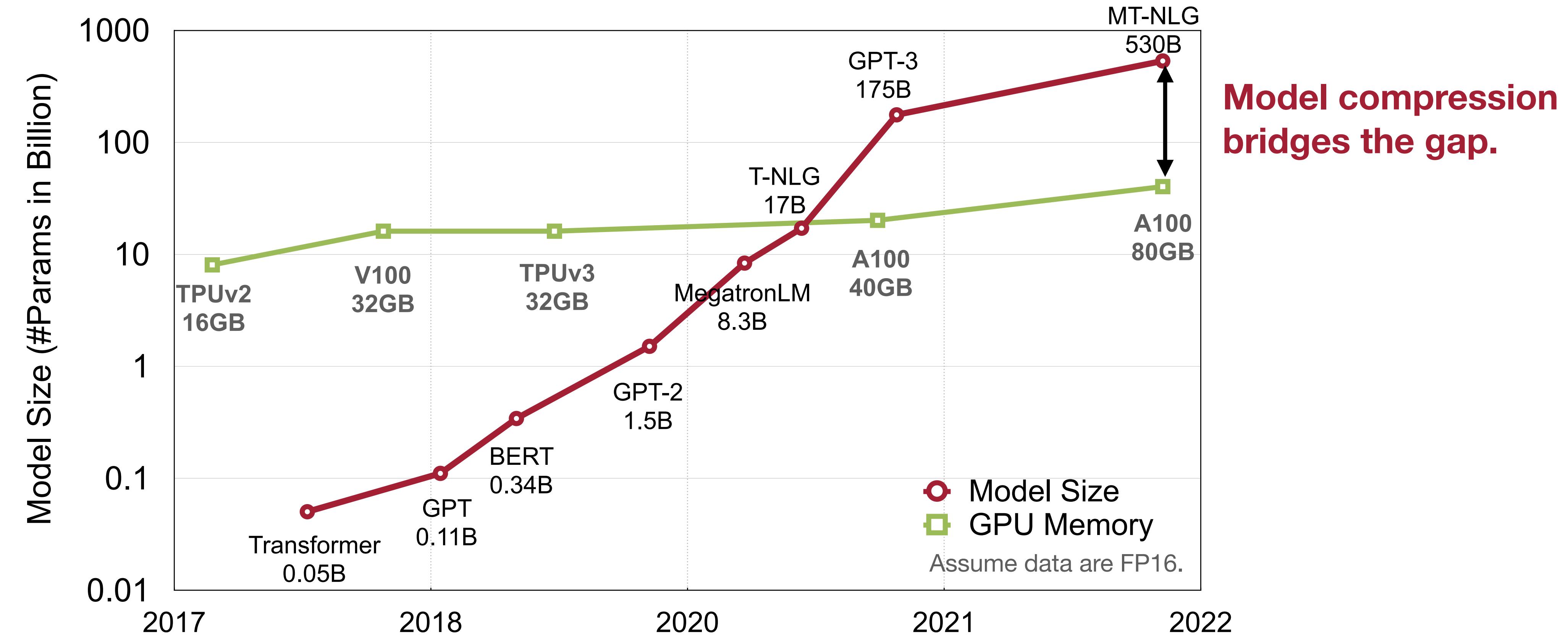
- Associate Professor, MIT EECS
- AWQ (best paper award of MLSys)
- MIT Technology Review: 35 Innovators under 35
- NSF Career Award
- IEEE “AI’s 10 to Watch: The Future of AI” Award
- Sloan Research Fellowship



- Cofounder of OmniML (now part of NVIDIA)

Model Compression and Efficient AI

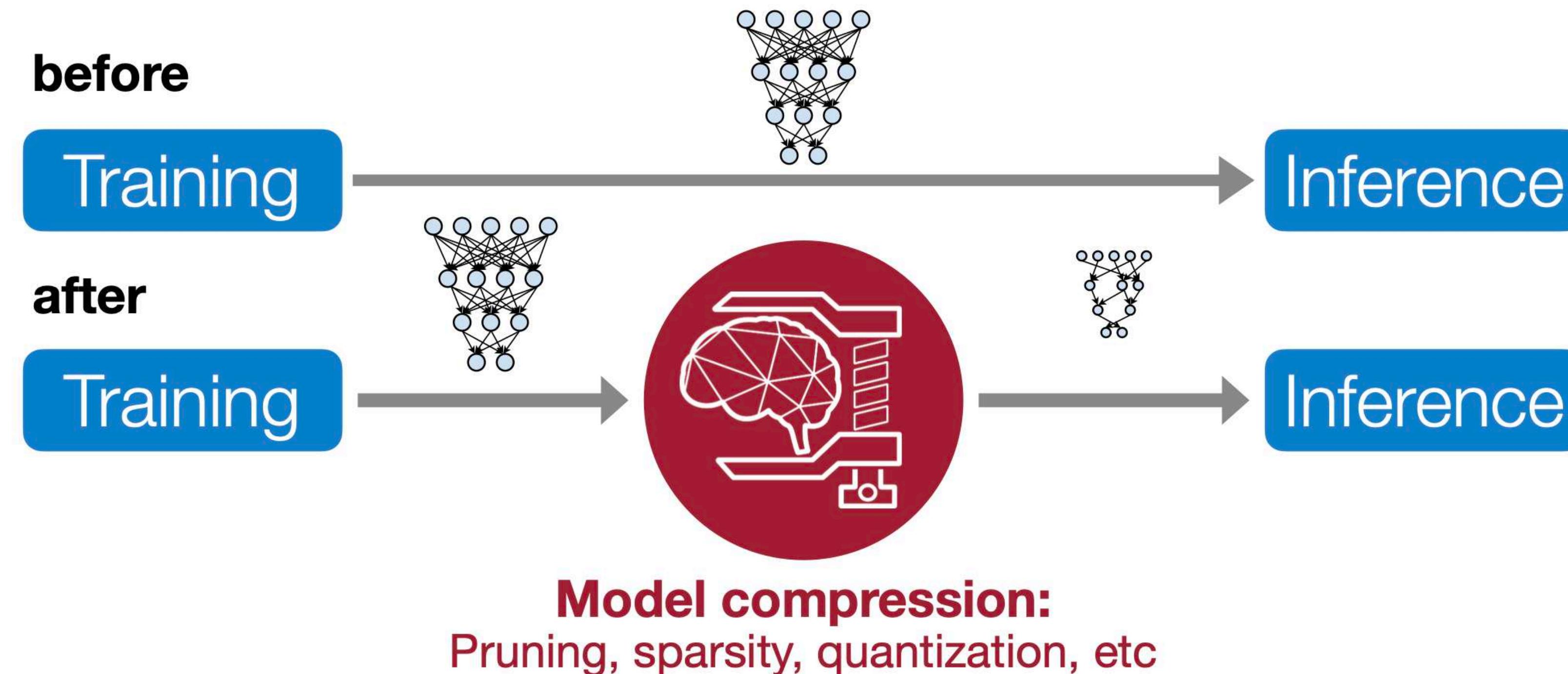
Bridges the Gap between the Supply and Demand of AI Computing



SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models (ICML 2023)

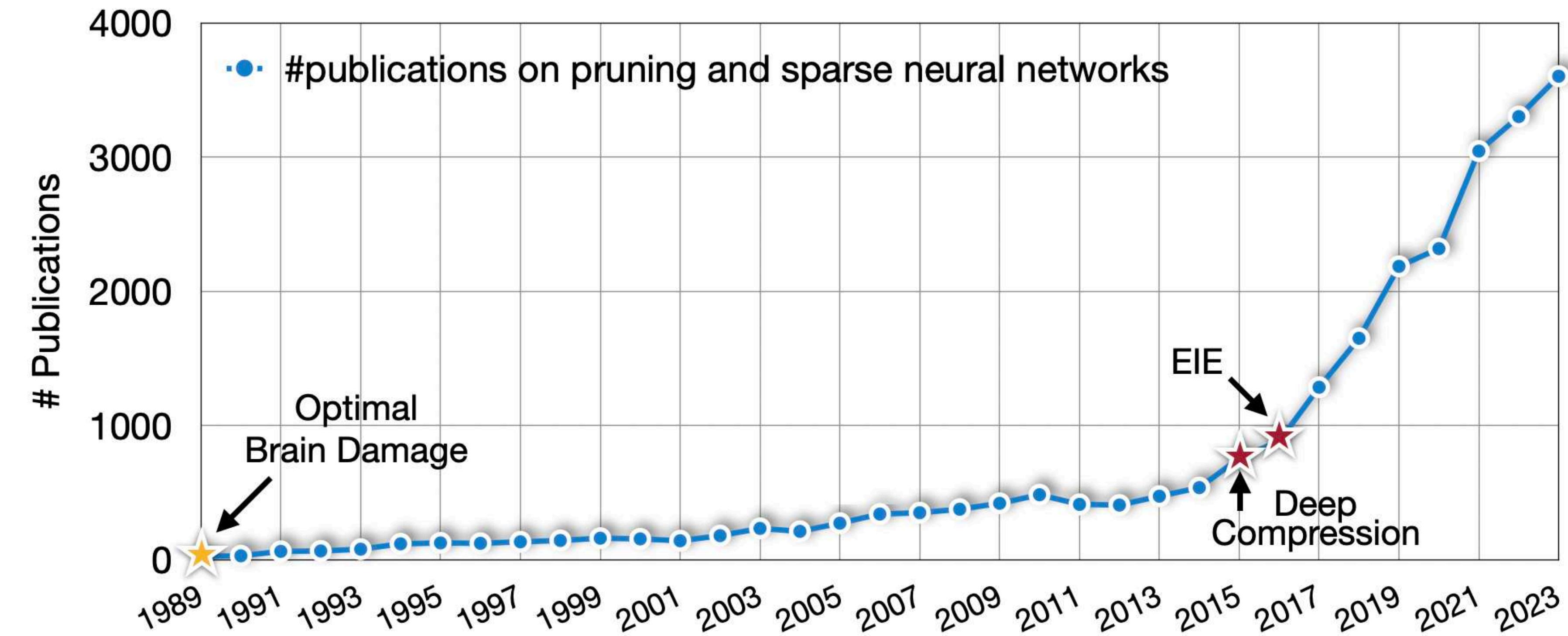
Model Compression and Efficient AI

Bridges the Gap between the Supply and Demand of AI Computing



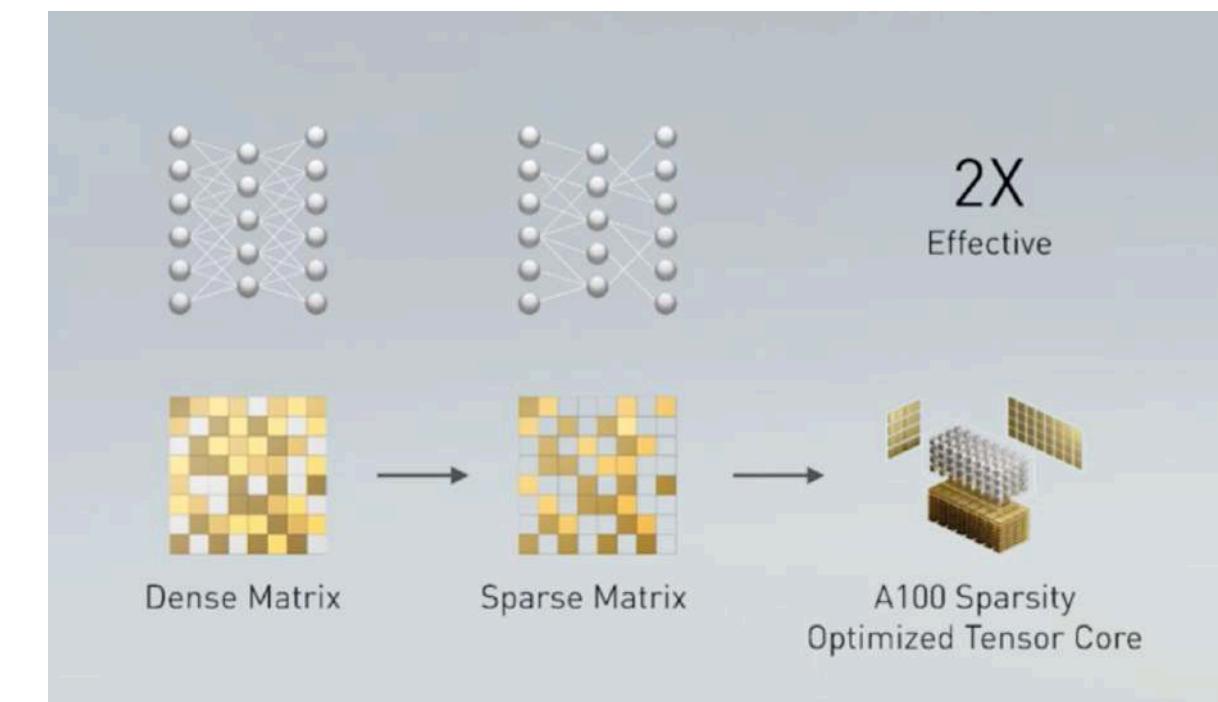
“Deep Compression” and EIE opened a new opportunity to build hardware accelerator for sparse and compressed neural networks

Pruning and Sparsity:



Quickly increased since 2015, including both algorithms and systems.
2:4 sparsity is supported in NVIDIA Ampere Sparse Tensor Core

Source: <https://github.com/mit-han-lab/pruning-sparsity-publications>



Deep Learning is Everywhere

But they are computationally costly

How to make them light-weighted and fast?



Image source: [1](#), [2](#), [3](#)

Deep Learning is Everywhere

But they are computationally costly

How to make them light-weighted and fast?



Image source: [1](#), [2](#), [3](#)

Deep Learning for Image Classification

DNNs achieve super-human classification accuracy on ImageNet

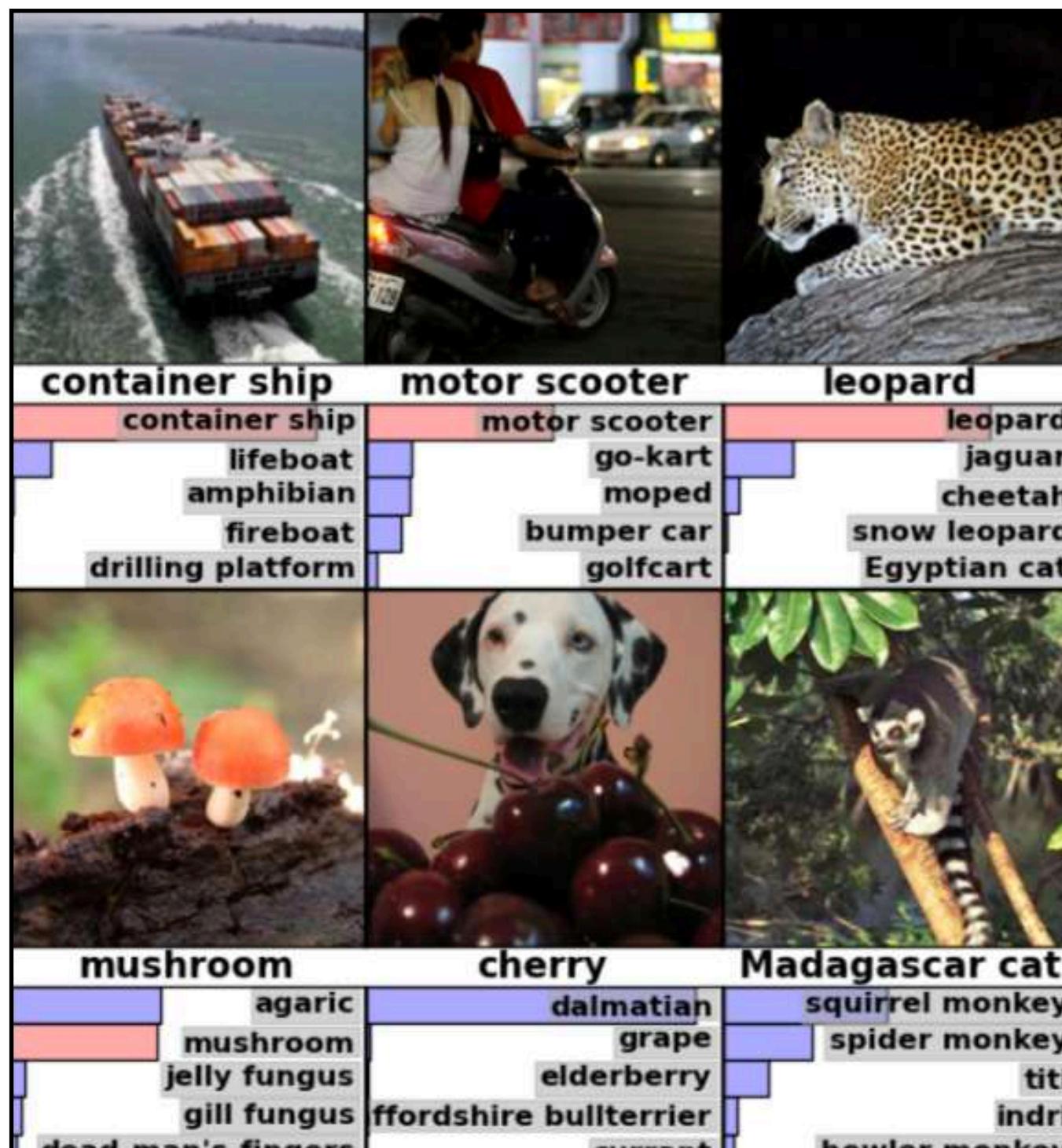
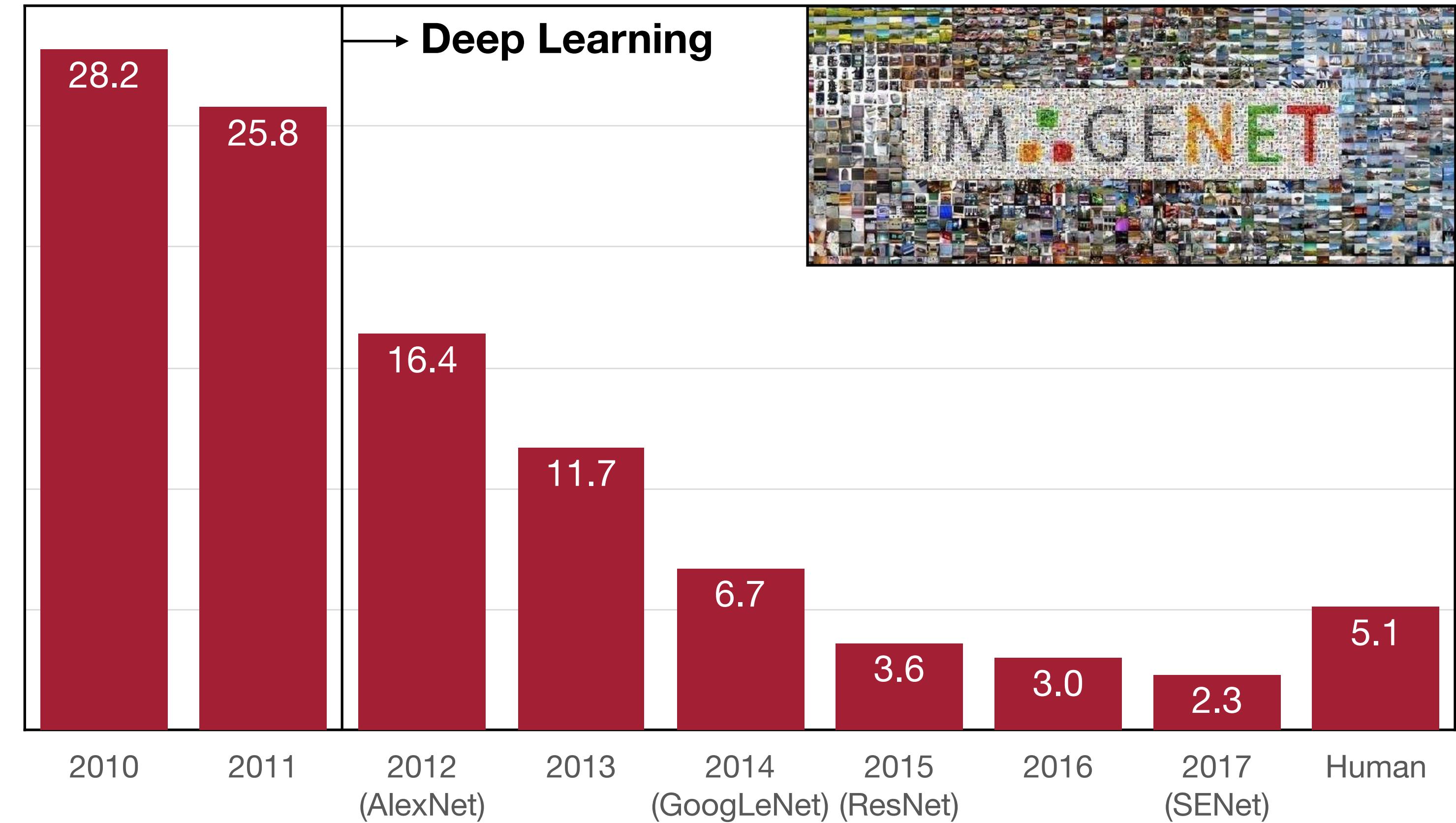


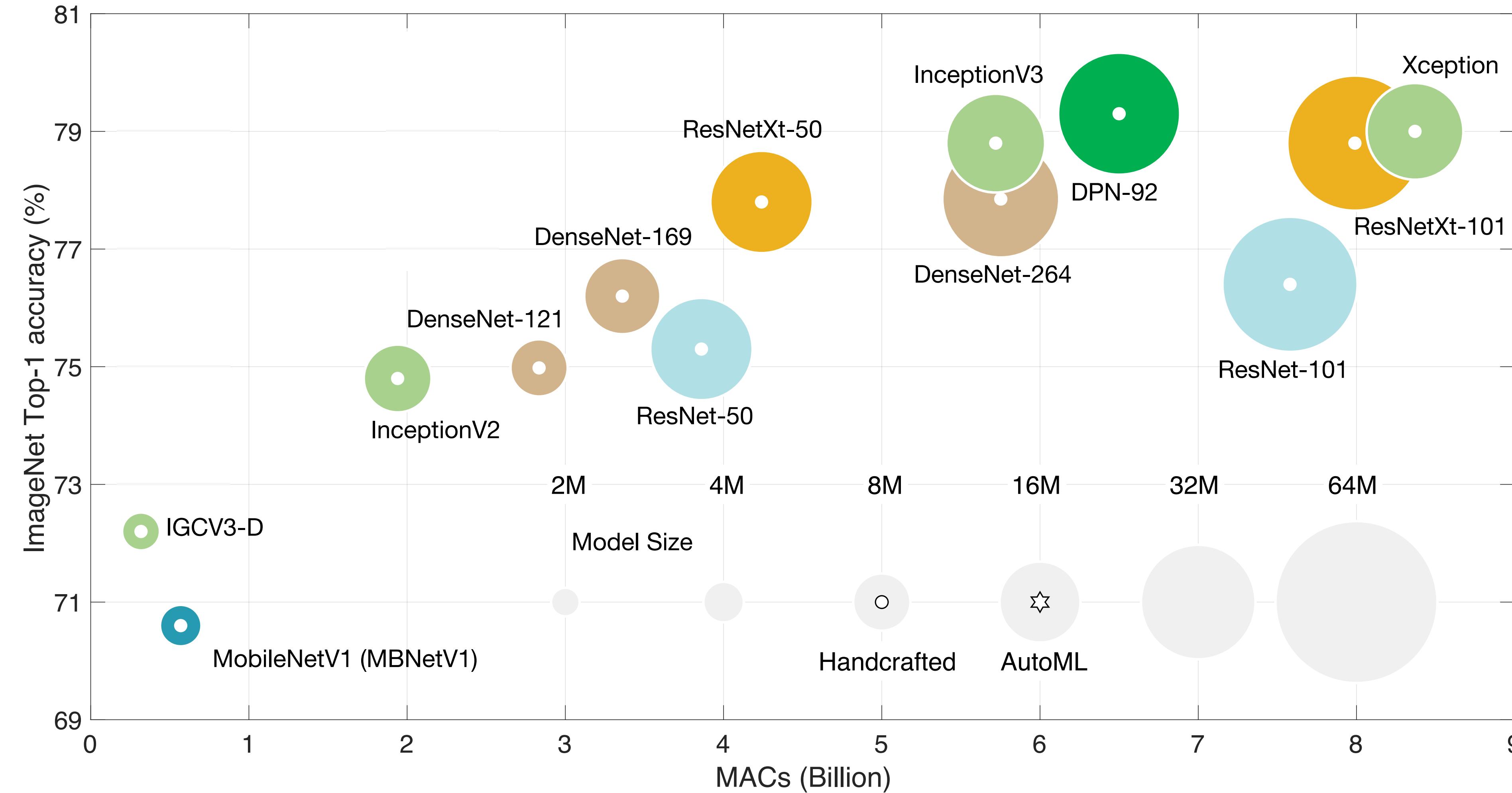
Image Classification

ImageNet Contest Winning Entry: Top 5 Error Rate (%)



Efficient Image Classification

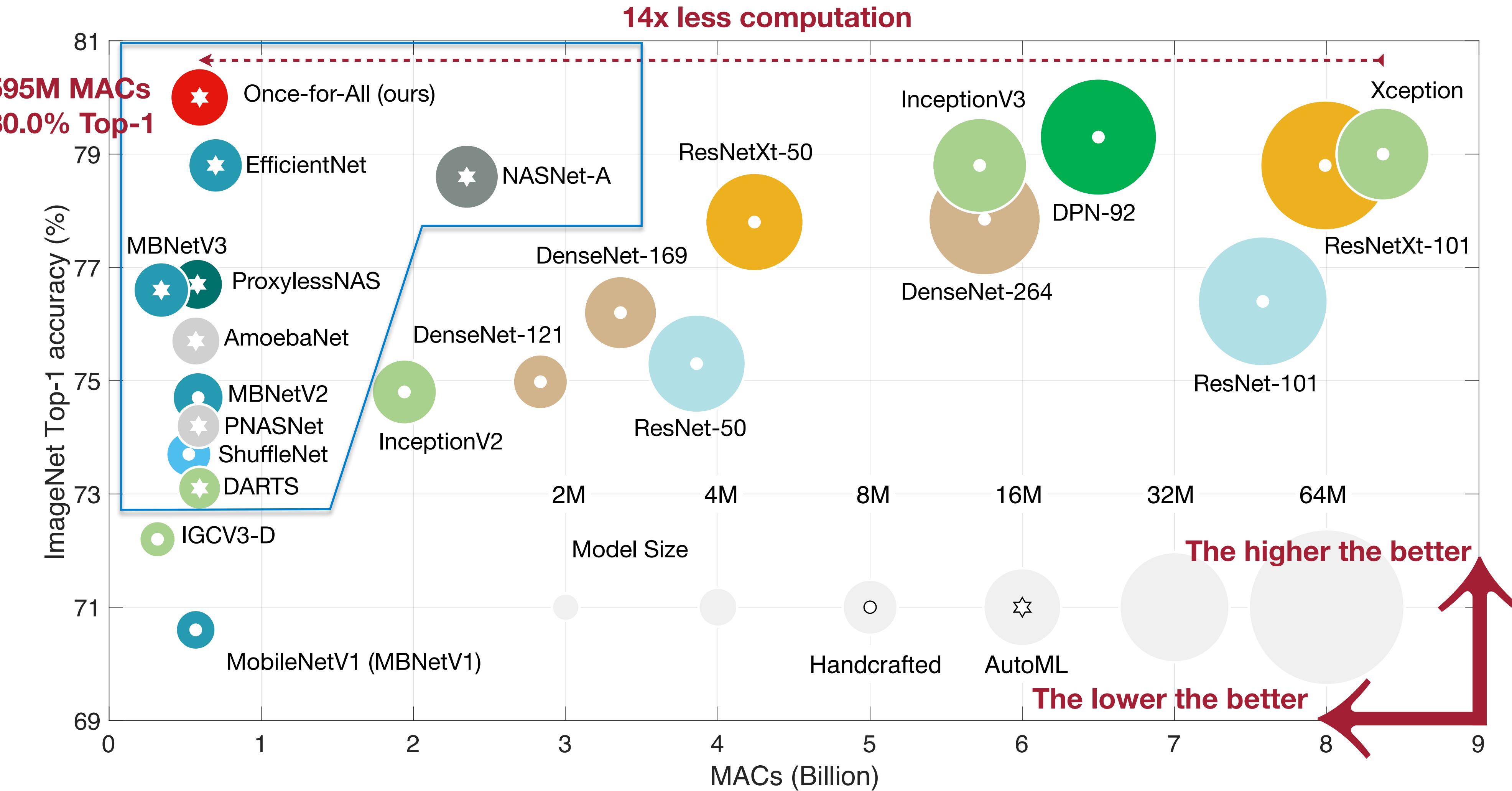
High accuracy comes at the cost of high computation



Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey [Deng et al., IEEE 2020]

Efficient Image Classification

Neural architecture search reduces the computational cost



Once-for-All: Train One Network and Specialize it for Efficient Deployment [Cai et al., ICLR 2020]

Efficient Image Classification

Efficient deep learning enables daily life application on mobile phones

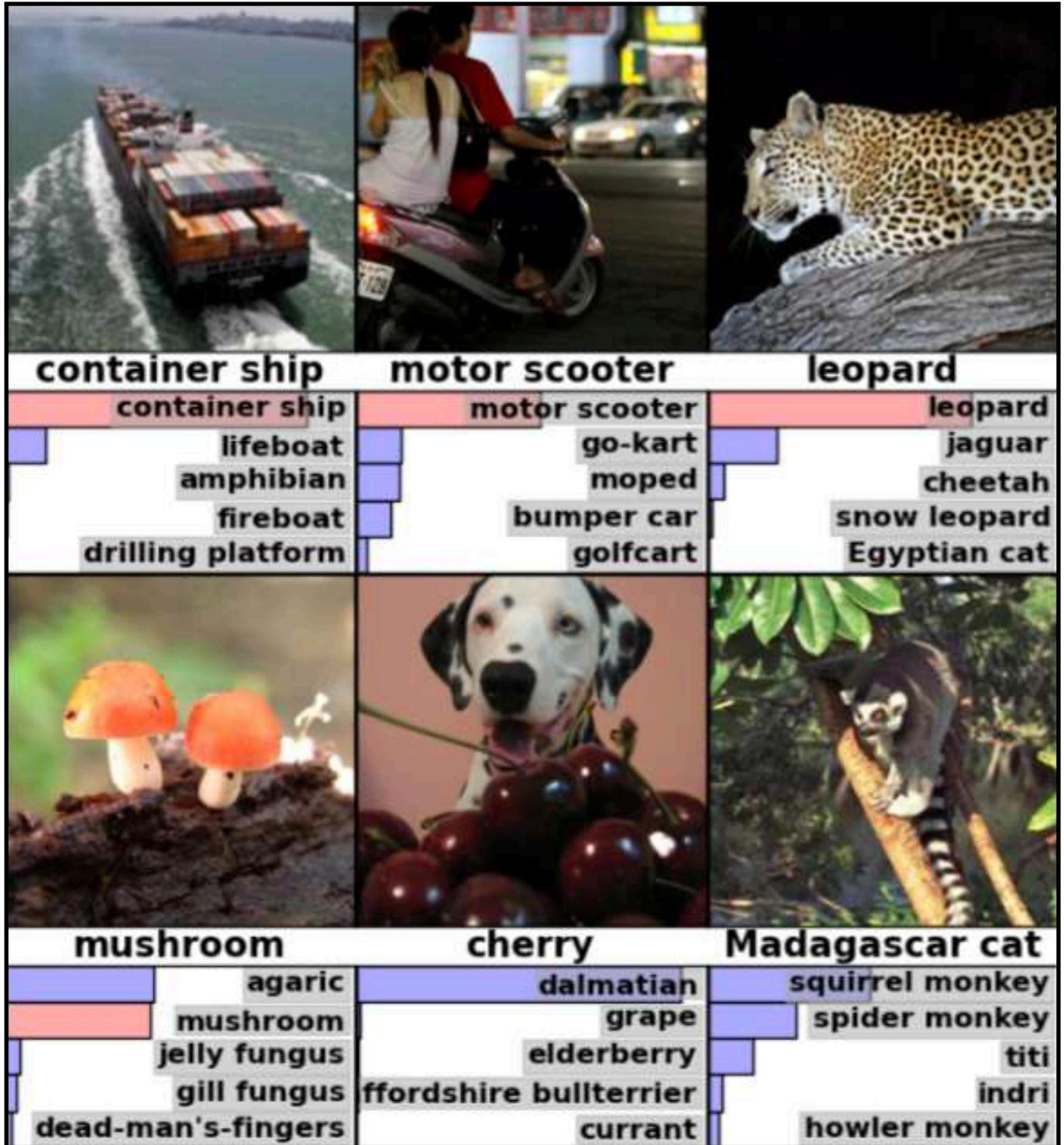


Image Classification

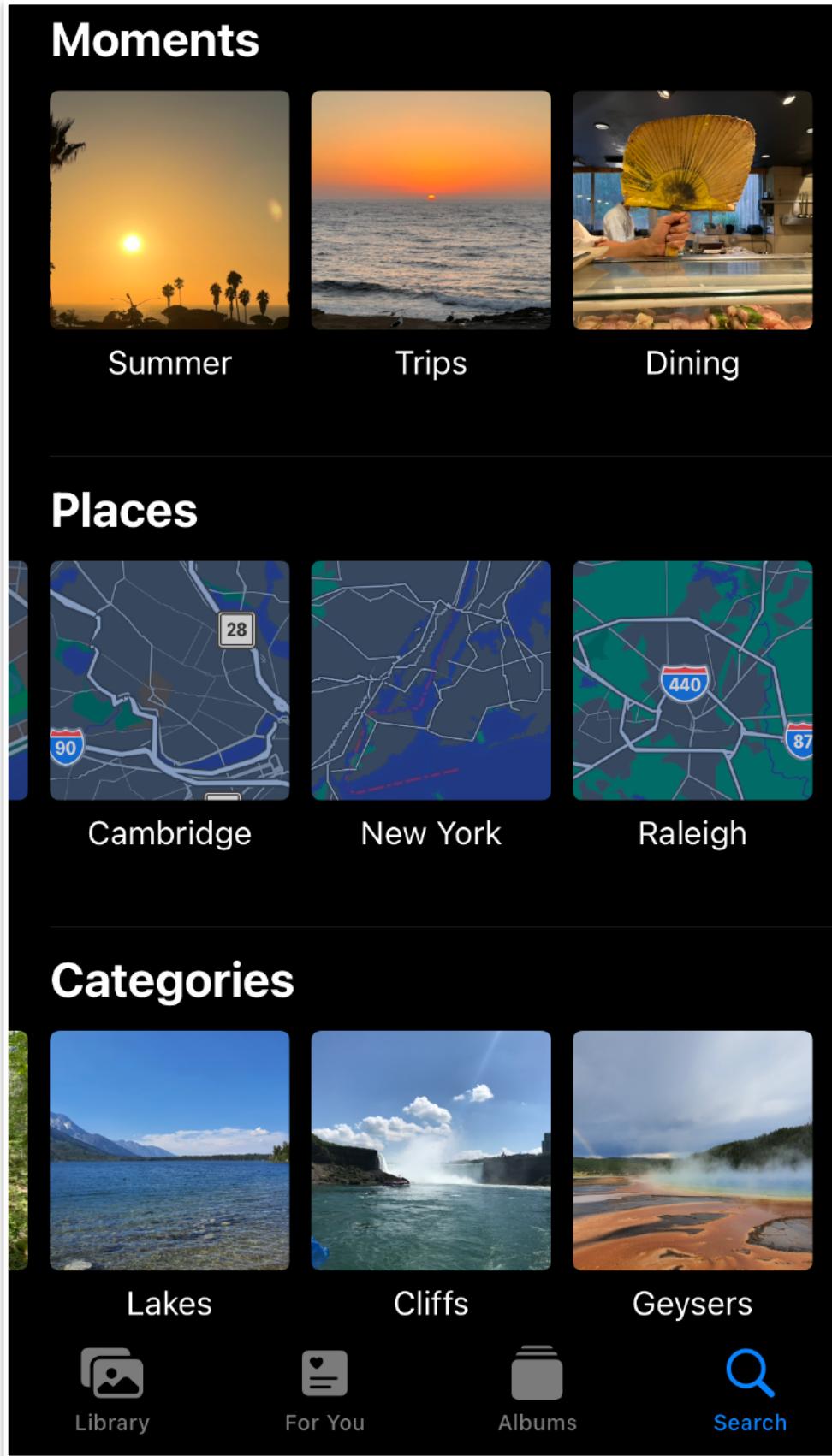
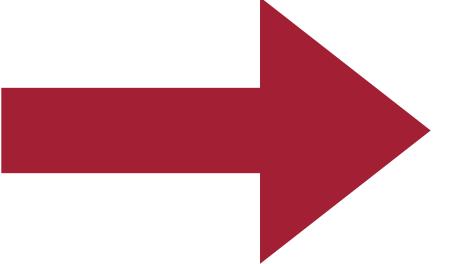
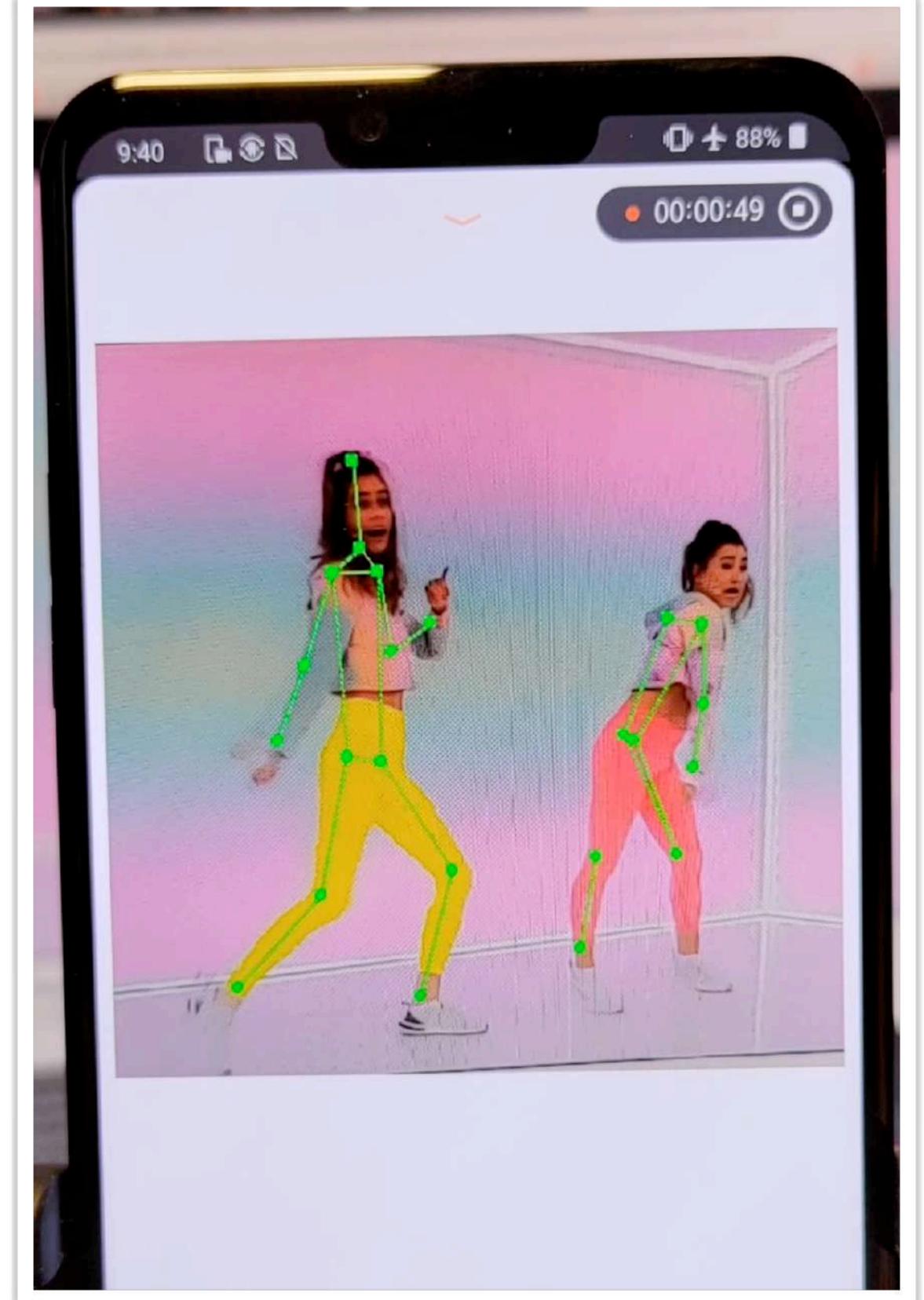
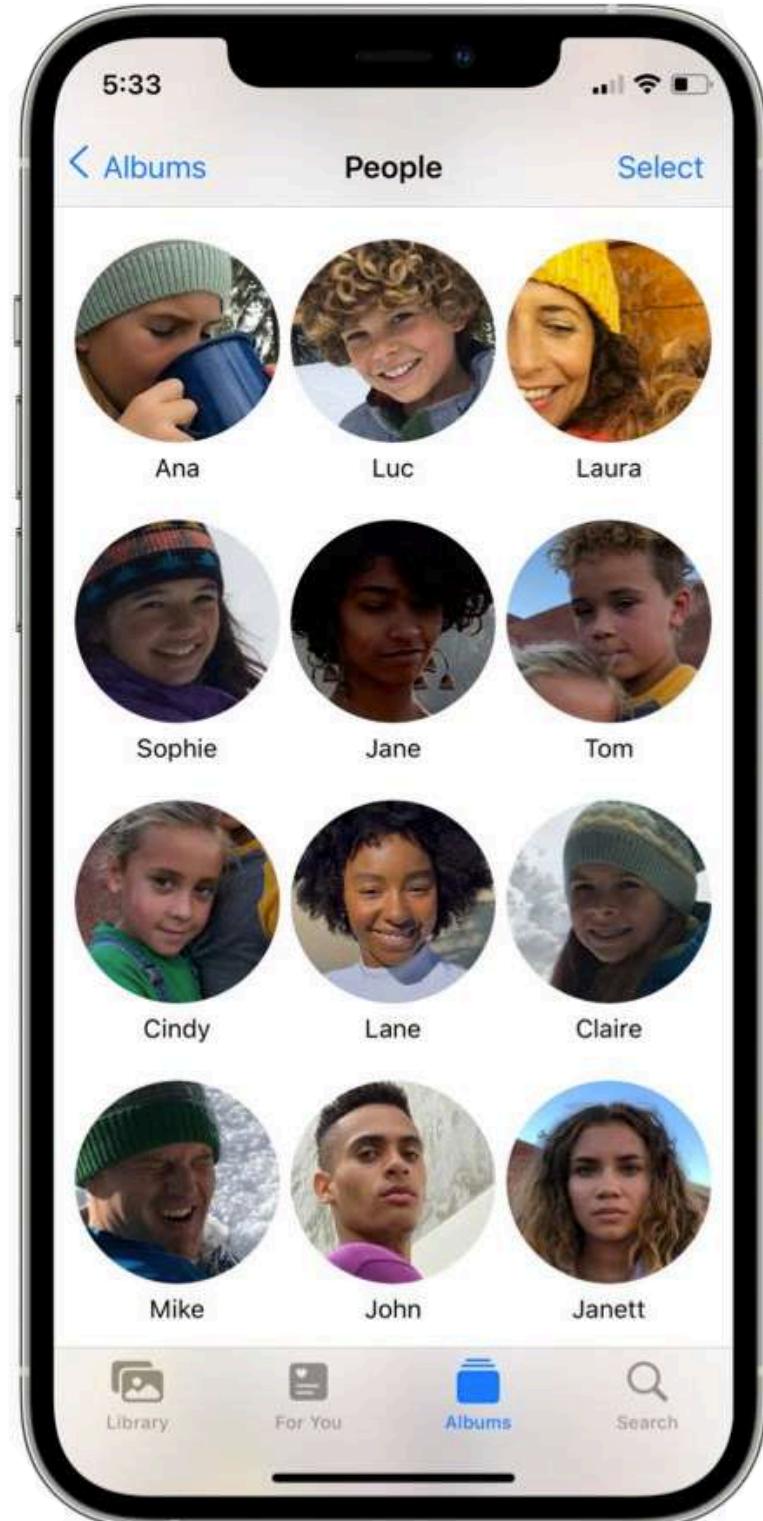


Photo Tags (on iPhone)

Efficient Image Recognition on Phones

Efficient deep learning enables daily life application on mobile phones



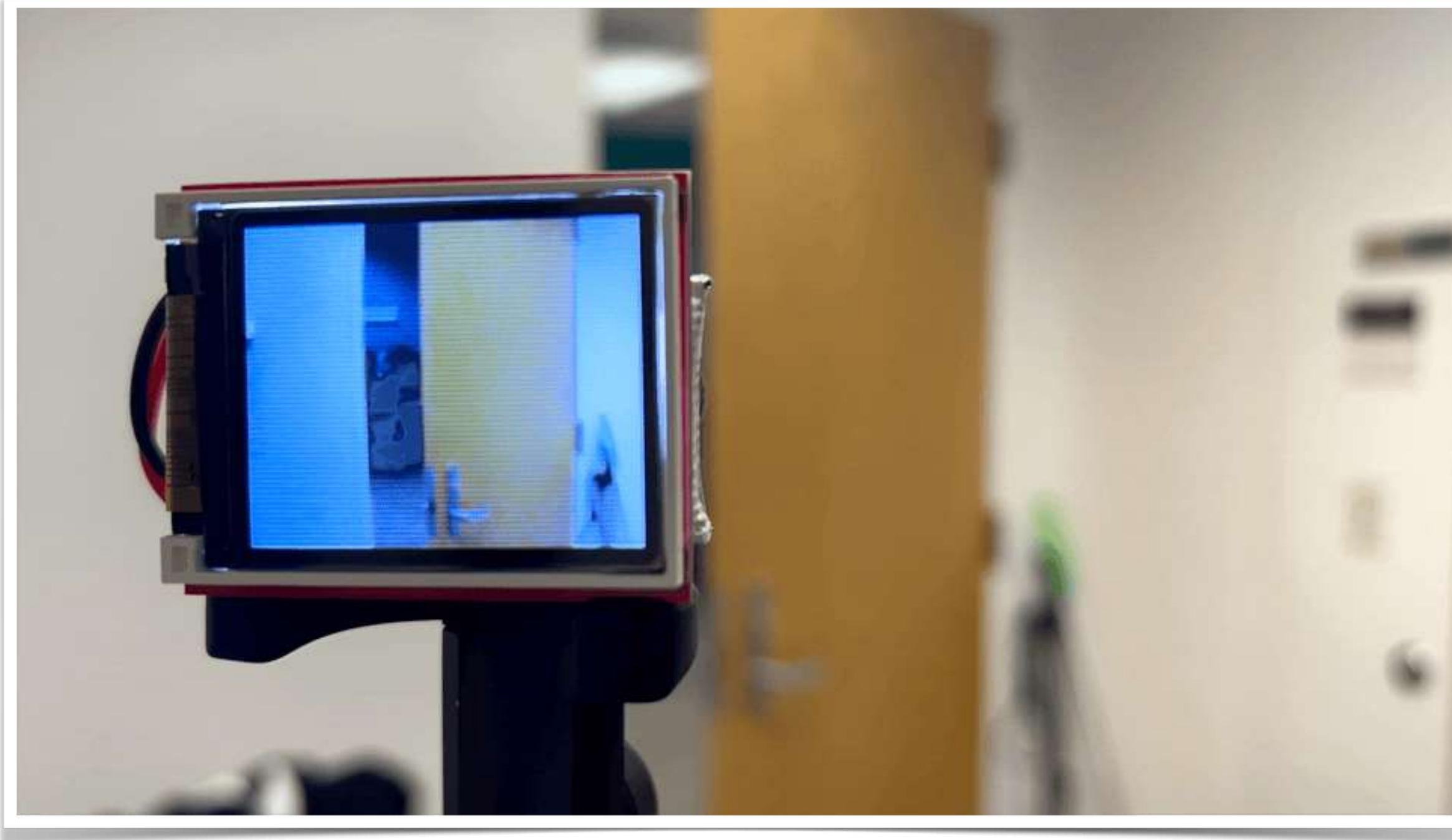
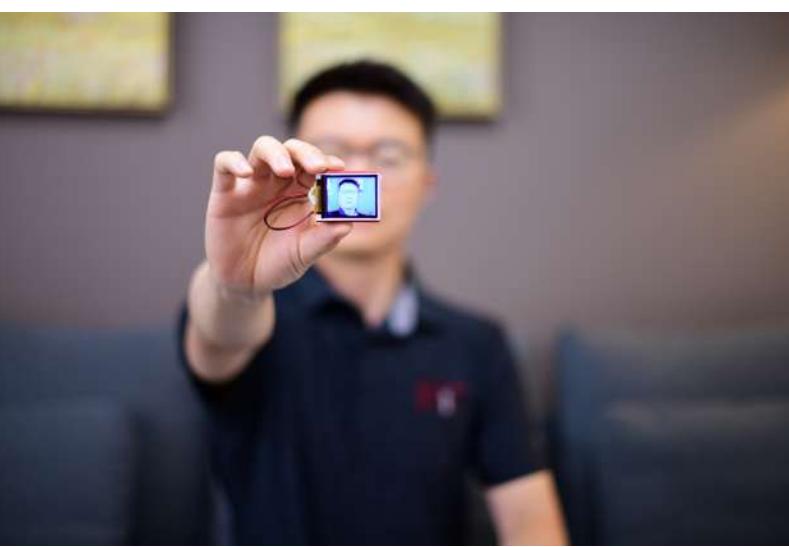
People Recognition (on iPhone)

On-Device Pose Estimation

Lite Pose: Efficient Architecture Design for 2D Human Pose Estimation [Wang et al., CVPR 2022]
Recognizing People in Photos Through Private On-Device Machine Learning [Apple, 2021]

Efficient Image Recognition on MCUs

MCUNet enables TinyML on IoT devices
(ARM Cortex-M7 + OpenMV Cam)



Facial Mask Detection



Person Detection

Efficient On-Device Training on Edge

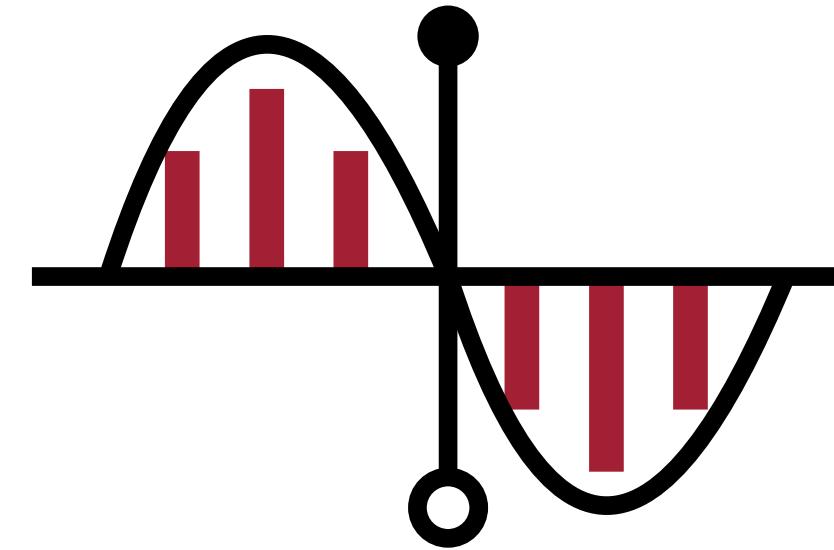
AI systems need to continually adapt to new data collected locally



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

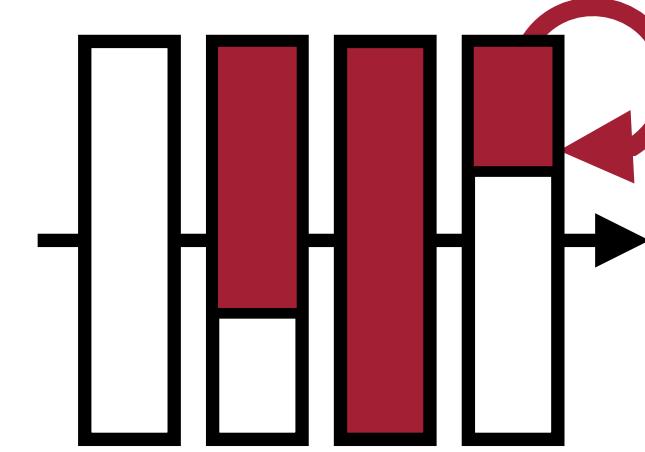
Efficient On-Device Training on MCUs

MCUNetV3 enables efficient on-device training on microcontrollers

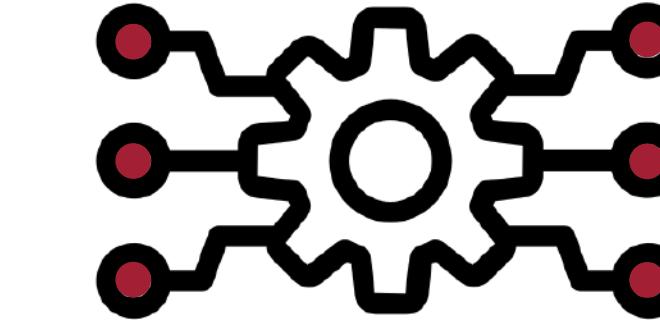


1. Quantization-aware scaling

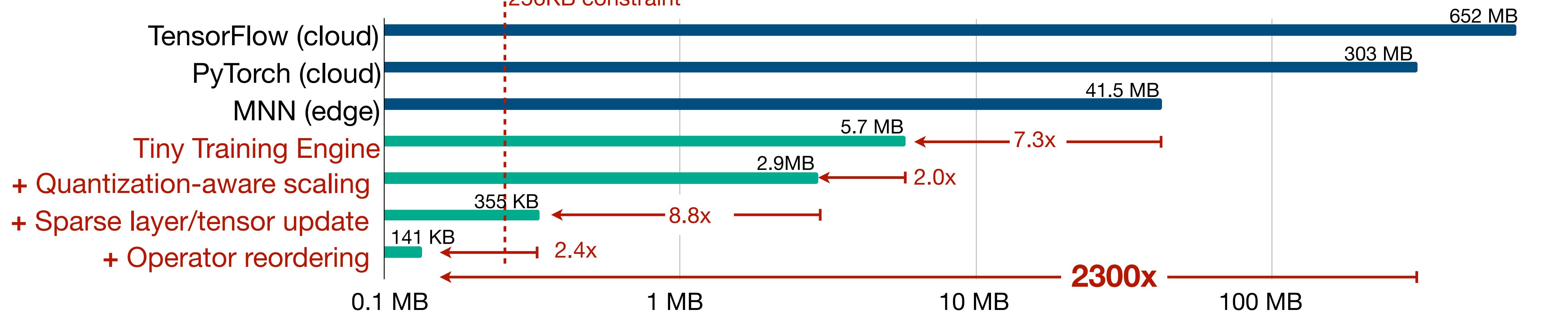
$$\tilde{G}_{\bar{W}} = G_{\bar{W}} \cdot s_{\bar{W}}^{-2}$$



2. Sparse layer/tensor update



3. Tiny Training Engine



On-Device Training Under 256KB Memory

Ji Lin*, Ligeng Zhu*, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, Song Han

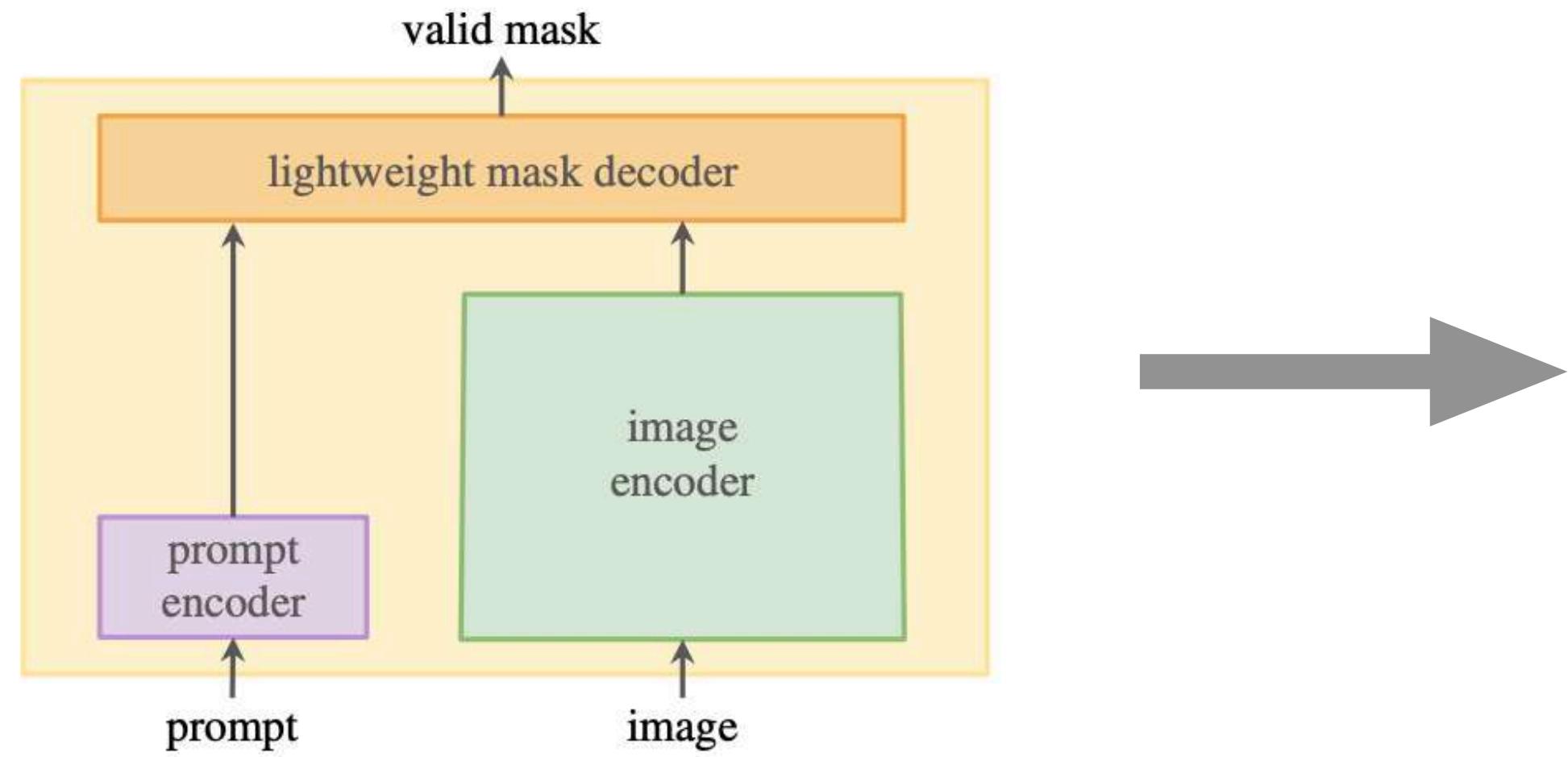
ImageNet Pre-trained MCUNet -> VWW

Running on OpenMV Cam H7 MCU

SRAM: 185KB / Flash: 467KB

Promptable Image Segmentation

Segment anything



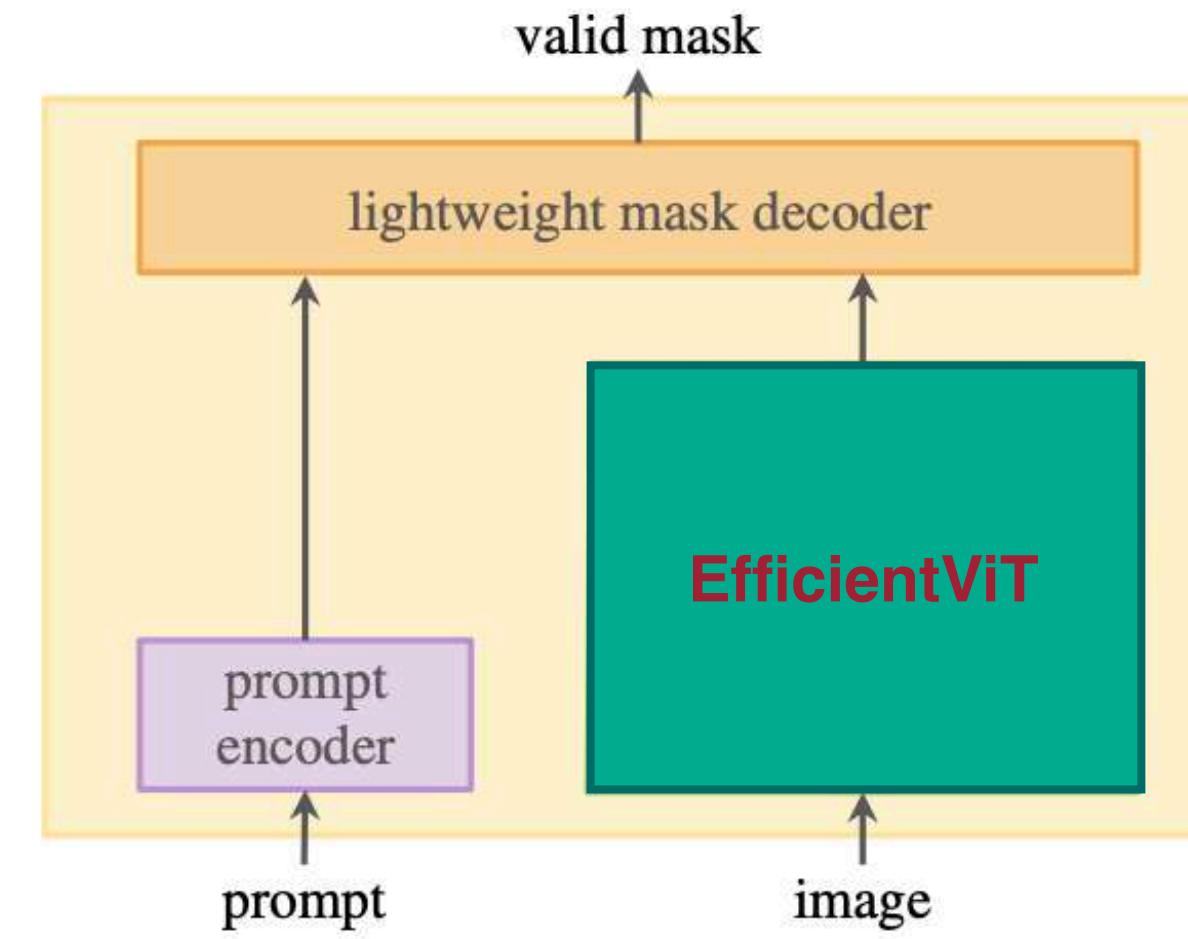
Promptable Image Segmentation Model



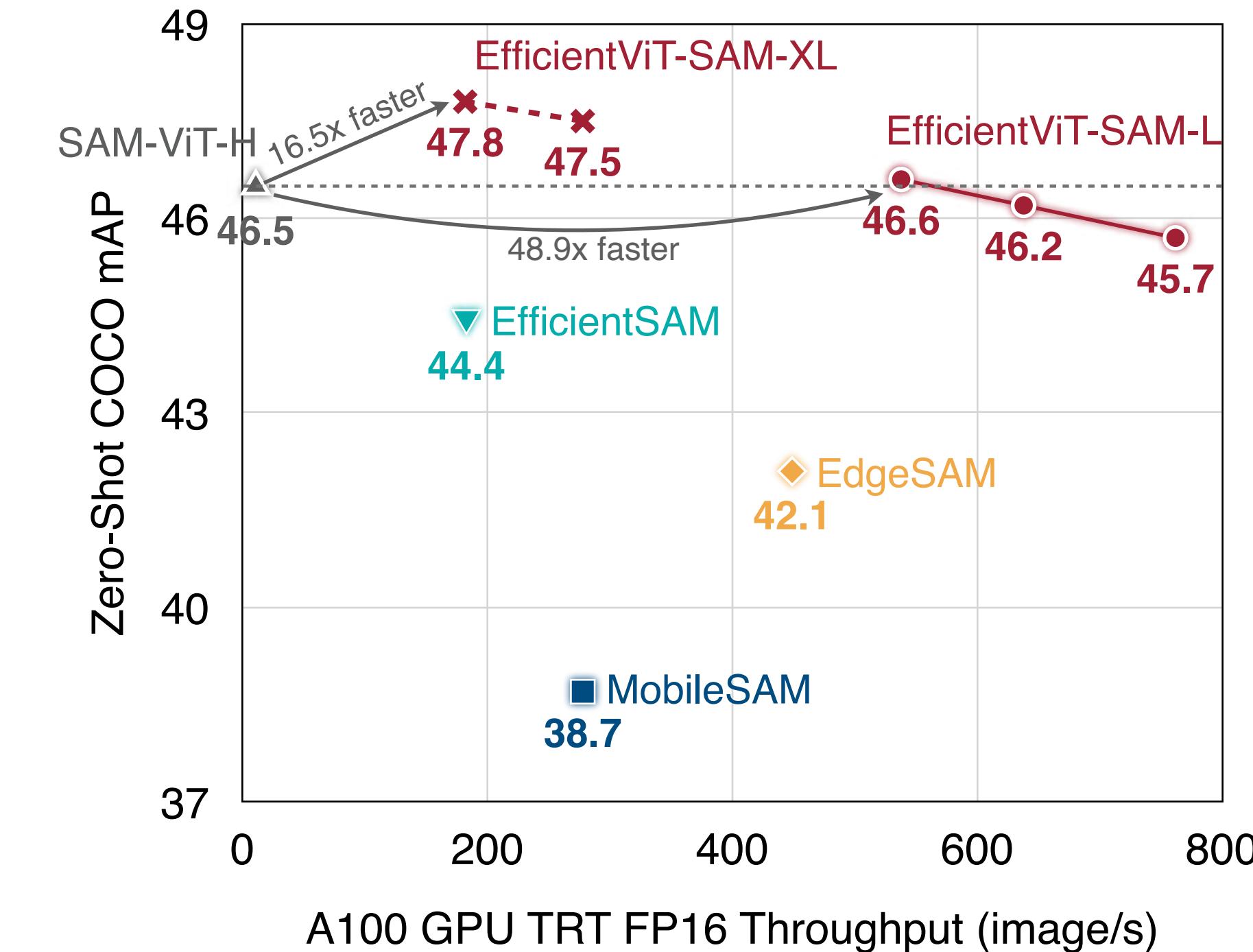
Segment Anything (Kirillov *et al.*, ICCV 2023)

EfficientViT-SAM

Accelerate segment anything with EfficientViT



Promptable Image Segmentation Model



- Match SAM-ViT-H's zero-shot segmentation performance with 48.9x speedup on A100 GPU.

EfficientViT: Multi-Scale Linear Attention for High-Resolution Dense Prediction [Cai et al., ICCV 2023]

EfficientViT-SAM

Accelerate segment anything with EfficientViT

- Online demo: <https://evitsam.hanlab.ai/>



EfficientViT: Multi-Scale Linear Attention for High-Resolution Dense Prediction [Cai et al., ICCV 2023]

From Discriminative Model to Generative Model

Diffusion models create realistic images from a natural language description

Teddy bears, mixing sparkling chemicals as mad scientists



A bowl of soup, as a planet in the universe, as a 1960s poster



A photo of an astronaut riding a horse on mars

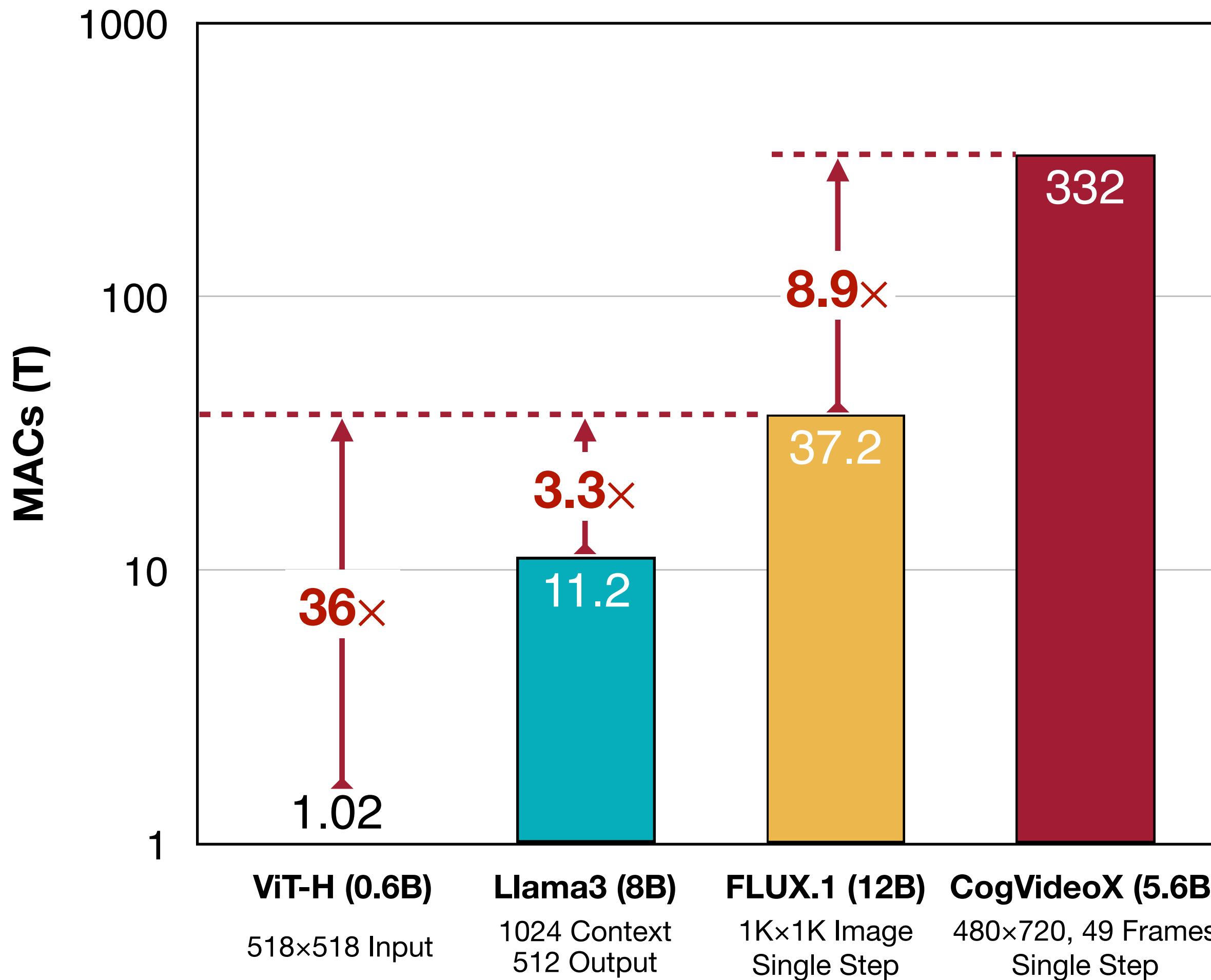


Training Stable Diffusion costs \$600,000 (256 A100s, 150k hours)

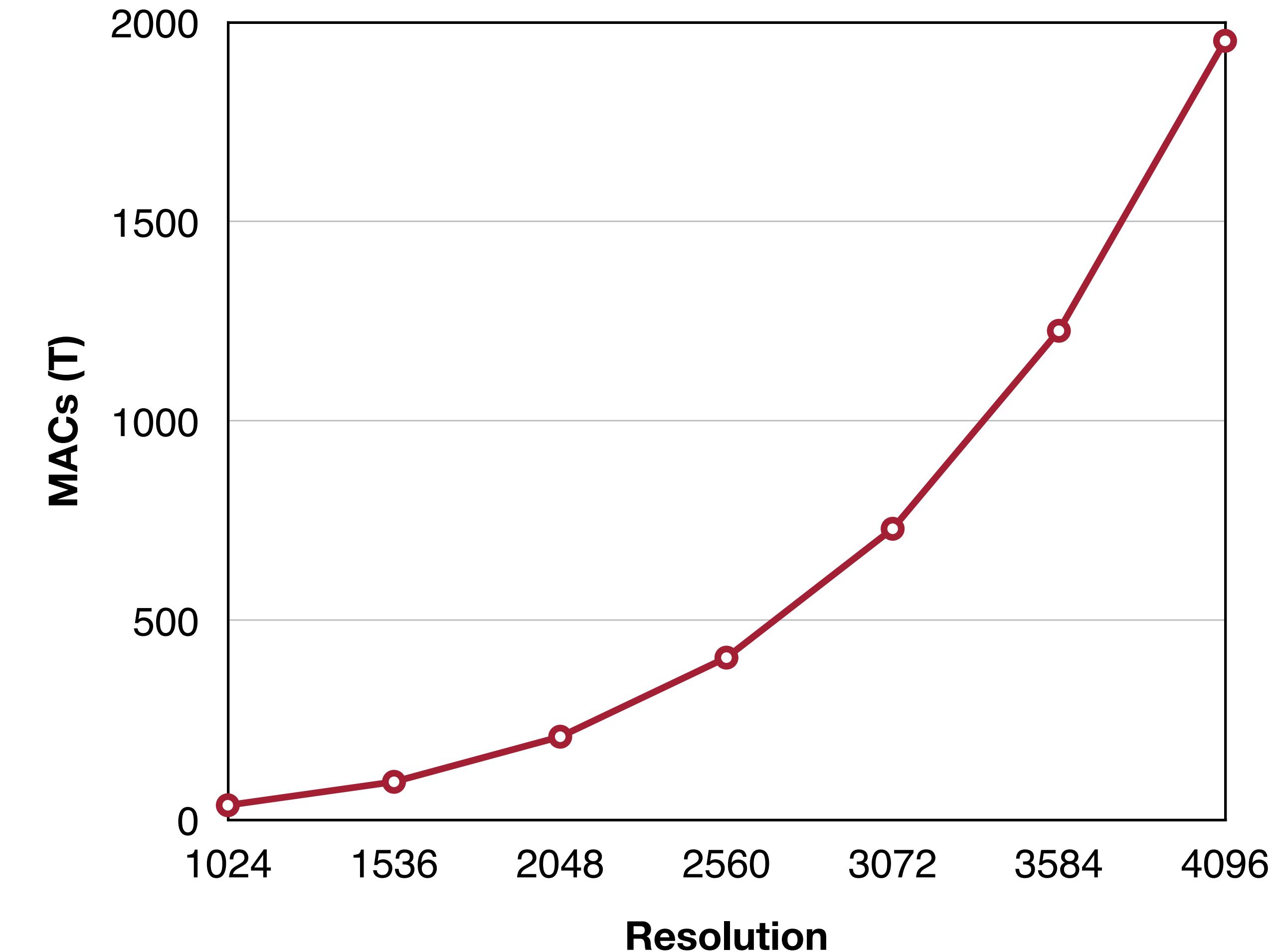
Midjourney: <https://www.midjourney.com/>

Generative models are computationally expensive

Comparisons with Other Models



Comparisons across Resolutions on FLUX.1



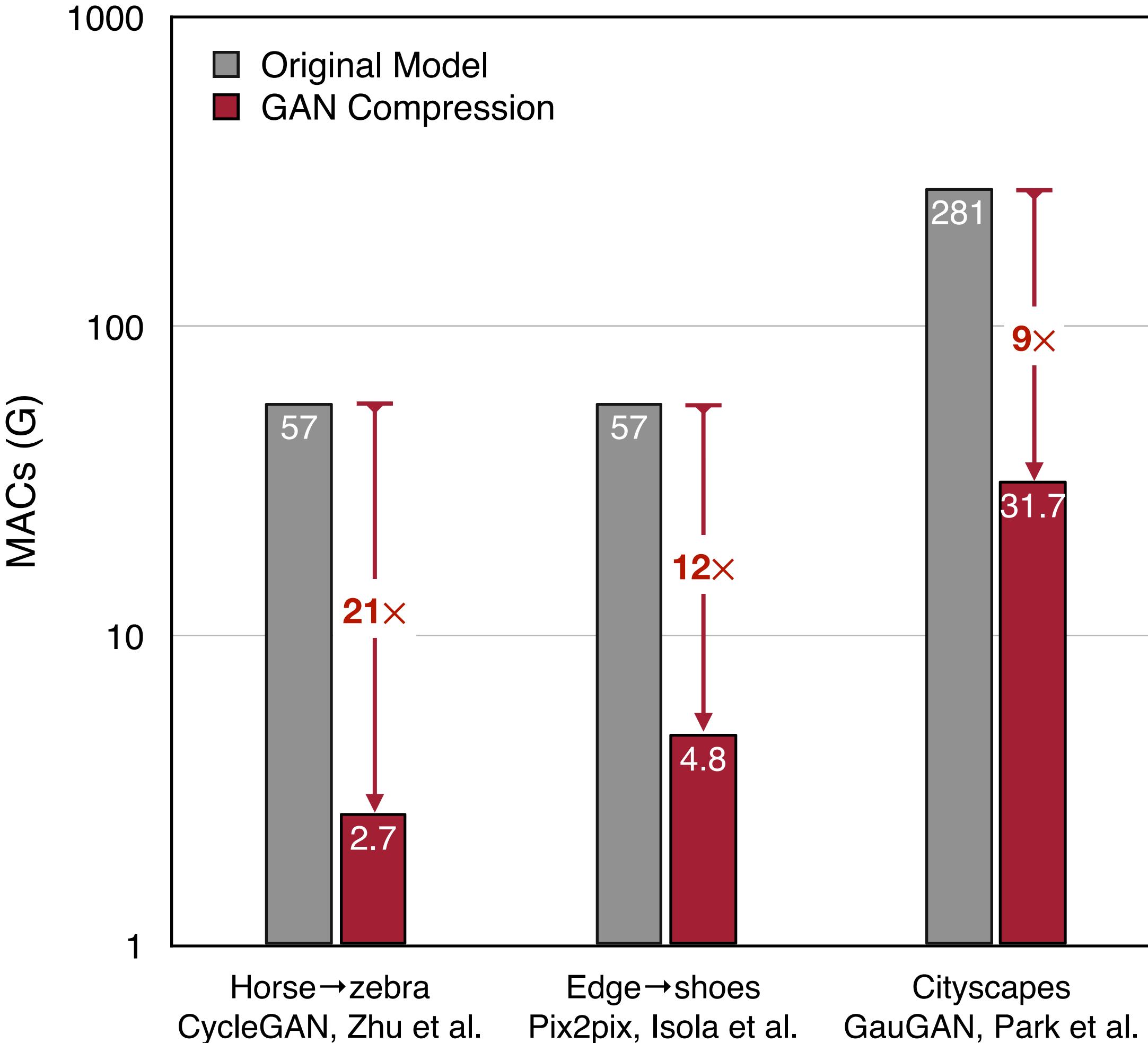
Diffusion models require more computation than other models. The computation scales super-quadratically with resolution.

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, ICLR 2021

CogVideoX: Text-to-Video Diffusion Models with An Expert Transformer

Efficient Image Generation

GAN Compression reduces the computation by 9-21X by pruning



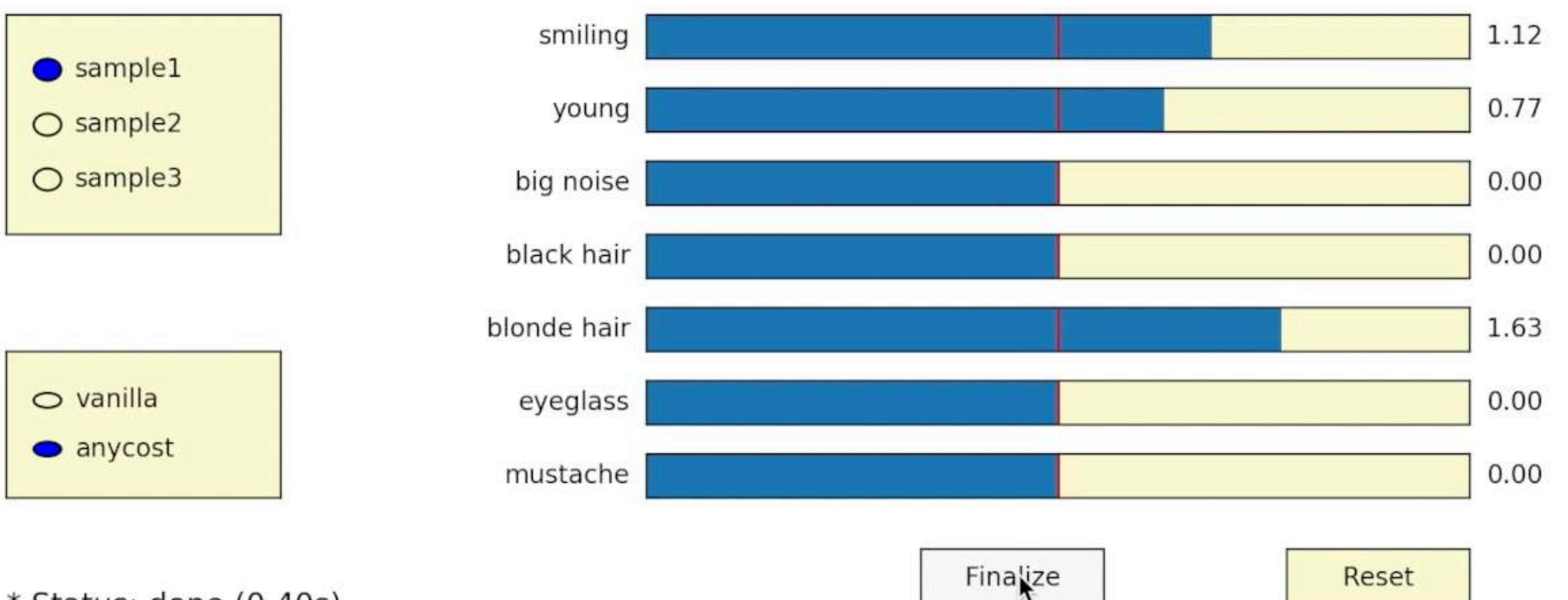
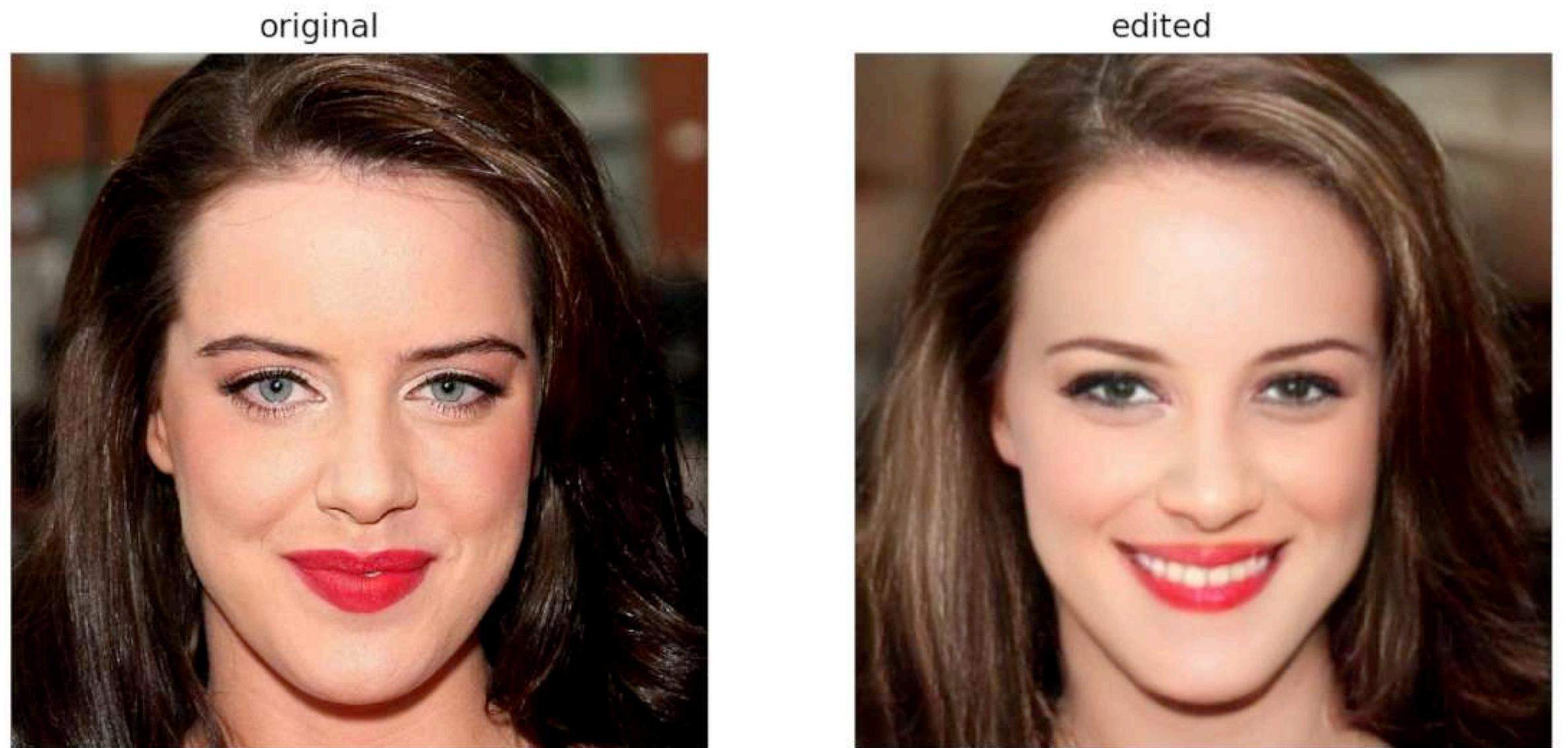
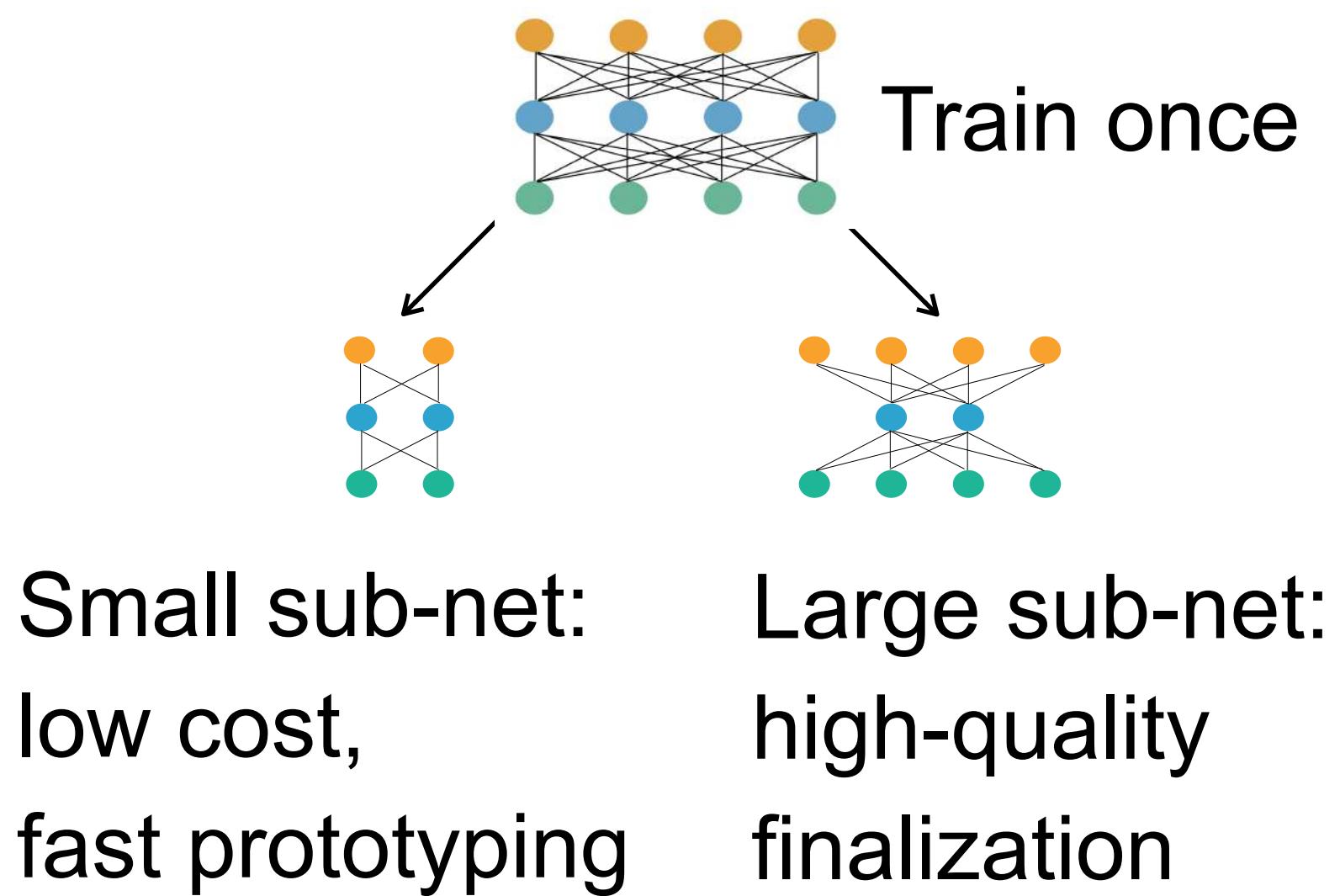
Measured on NVIDIA **Jetson Xavier GPU**
Lower FID indicates better Performance.



GAN Compression: Efficient Architectures for Interactive Conditional GANs [Li et al., CVPR 2020]

Efficient Image Generation

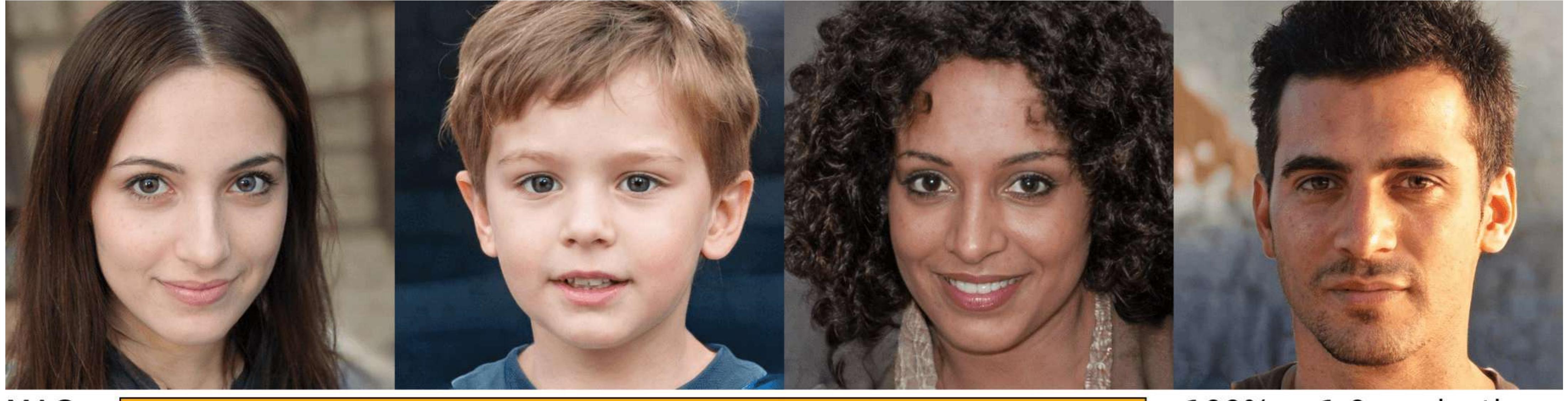
- Generative Adversarial Network (GAN) is computationally heavy and slow
- Difficult for interactive photo editing on mobile device (iPad)
- Anycost GAN with once-for-all network:



[Anycost GAN, CVPR'21](#)

Efficient Image Generation

AnycostGAN achieves interactive image synthesis and editing on a laptop



MACs:  100% 1.0x reduction

| Compute Budget | 1x | 0.7x | 0.5x | 0.4x | 0.2x |
|-----------------------|--------|---------|---------|---------|---------|
| Tiered Pricing | \$0.01 | \$0.007 | \$0.005 | \$0.004 | \$0.002 |

The quality is still reasonably good

Anycost GANs for Interactive Image Synthesis and Editing [Lin et al., CVPR 2021]

Efficient Image Generation

SIGE accelerates Stable Diffusion by >4X with spatial sparsity

A photograph of a horse on a grassland.



Original



11.6% Masked



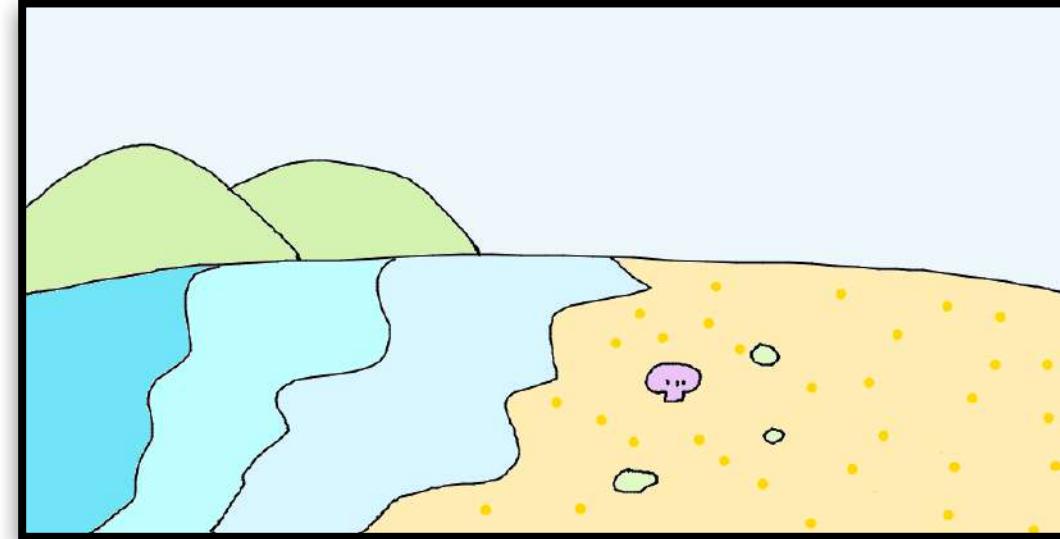
Stable Diffusion:
1855GMACs 369ms

Image Inpainting

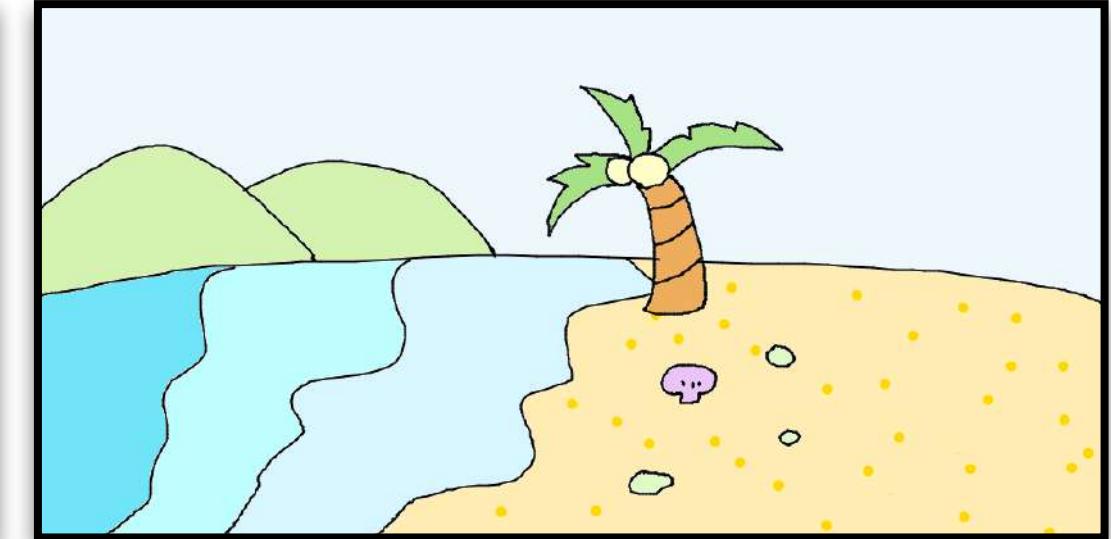


Ours:
514G (3.6×) 95.0ms (3.9×)

A fantasy beach landscape, trending on artstation.



Original



2.9% Edited



Stable Diffusion+SDEdit:
1855GMACs 369ms

Image Editing



Ours:
353G (5.3×) 76.4ms (4.8×)

Latency Measured on NVIDIA RTX 3090



Efficient Spatially Sparse Inference for Conditional GANs and Diffusion Models [Li et al., NeurIPS 2022]

Efficient Image Generation

DistriFusion accelerates high-resolution generation with distributed parallelism.

Original, 1 GPU
MACs: 907T
Latency: **12.3s**



Naïve Parallelization, 4 GPUs
MACs Per Device: 190T (4.8x Less)
Latency: 3.14s (3.9x Faster)
But w/ Artifact: Duplicated Subjects



DistriFusion (Ours), 4 GPUs
MACs Per Device: 227T (**4.0x Less**)
Latency: **4.16s (3.0x Faster)**
w/o Artifacts



Ethereal fantasy concept art of an elf, magnificent, celestial, ethereal, painterly, epic, majestic, magical, fantasy art, cover art, dreamy.



Romantic painting of a ship sailing in a stormy sea, with dramatic lighting and powerful waves.

Efficient Image Generation

FastComposer achieves tuning-free multi-subject image generation

- Create personalized images based on user-specified inputs

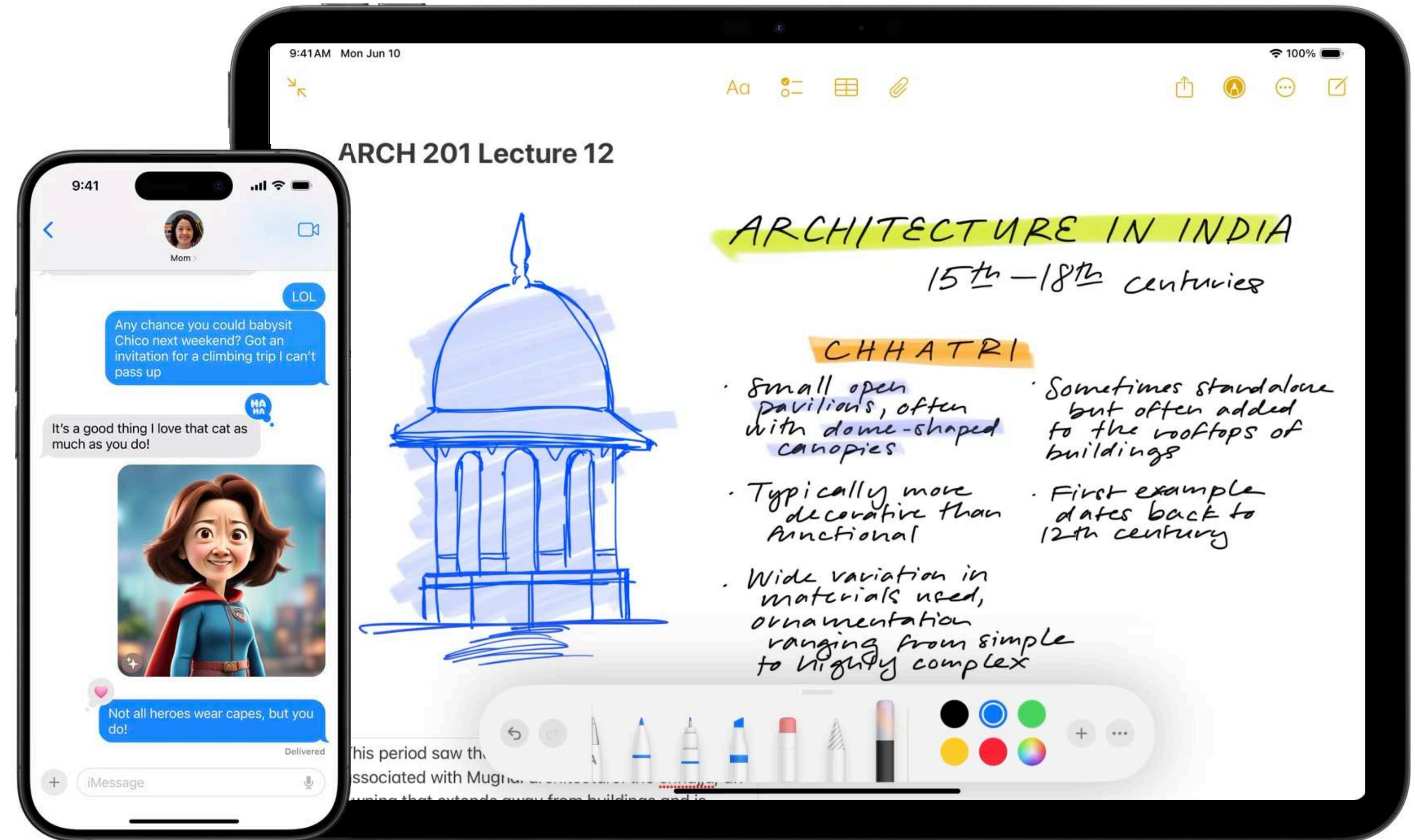


“Song Han riding a horse”
(Midjourney)

The screenshot shows the FastComposer interface. At the top, there are two input image boxes: "Image 1" containing a portrait of Jensen Huang and "Image 2" containing a portrait of Lady Gaga. Below these are two "Examples" boxes: one with a portrait of a man and another with a portrait of a woman. A central text field says "Upload the image for your subject". At the bottom left, a "Prompt" field contains the text "A man img and a woman img singing together". To the right, a "Job Status" box shows "run successfully". On the far right, a large generated image shows Jensen Huang and Lady Gaga singing together, with Lady Gaga holding a microphone. There are also five small thumbnail preview images at the bottom right.

FastComposer: <https://fastcomposer.hanlab.ai/>

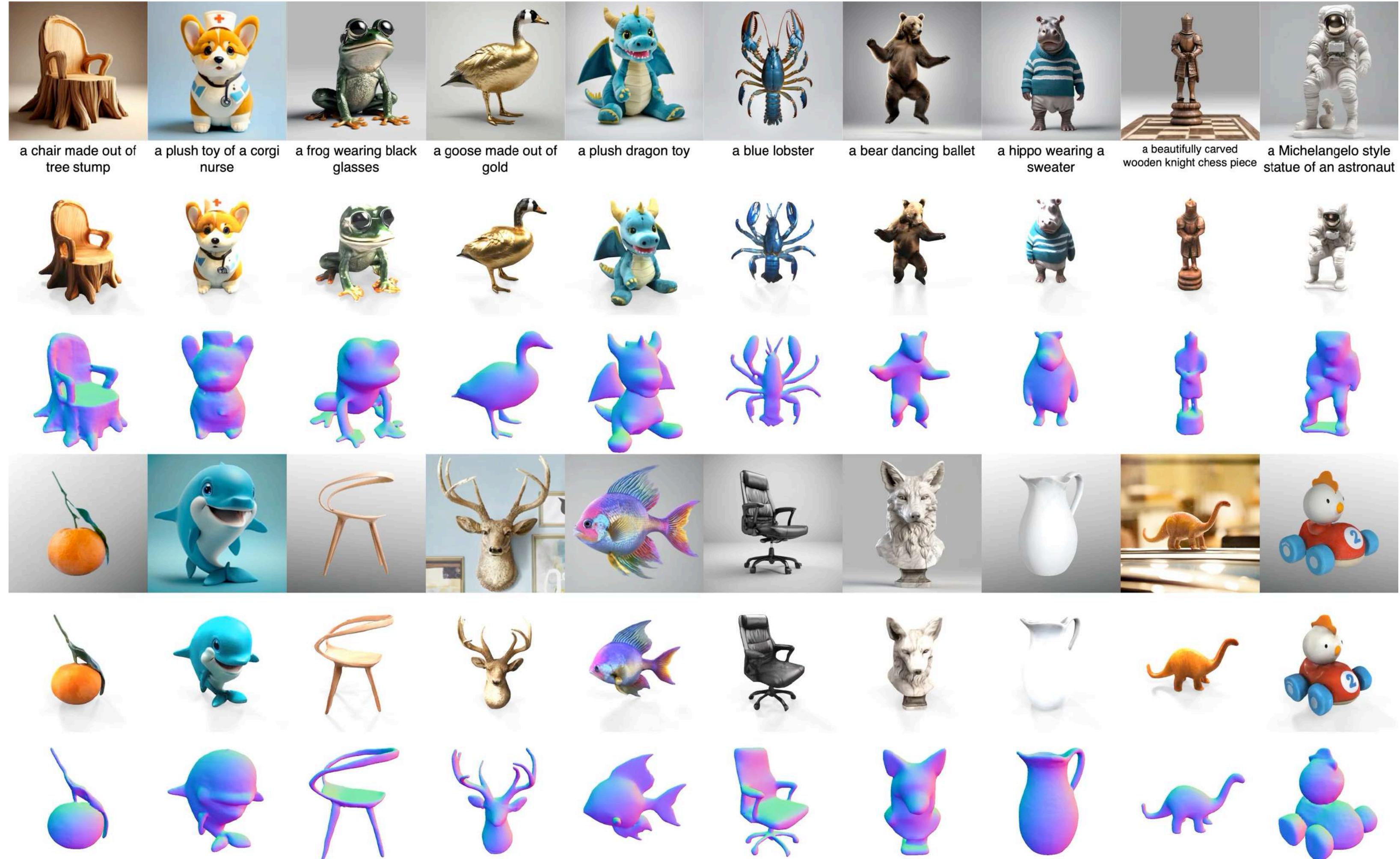
Efficient Image Generation



Source: <https://www.apple.com/apple-intelligence/>

3D Generation

Diffusion models create 3D objects from a reference image



One-2-3-45++: <https://sudo-ai-3d.github.io/One2345plus> page

3D Generation

Diffusion models create 3D objects from a natural language description



GSGen: <https://gsgen3d.github.io/>

Video Generation

Diffusion models create realistic videos from a natural language description



“Video modeling is a harder task for which performance is not yet saturated at **5.6B model size**”

Imagen Video: <https://imagen.research.google/video/>

Video Generation

Diffusion models create realistic videos from a natural language description



Prompt: A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She walks confidently and casually. The street is damp and reflective, creating a mirror effect of the colorful lights. Many pedestrians walk about.

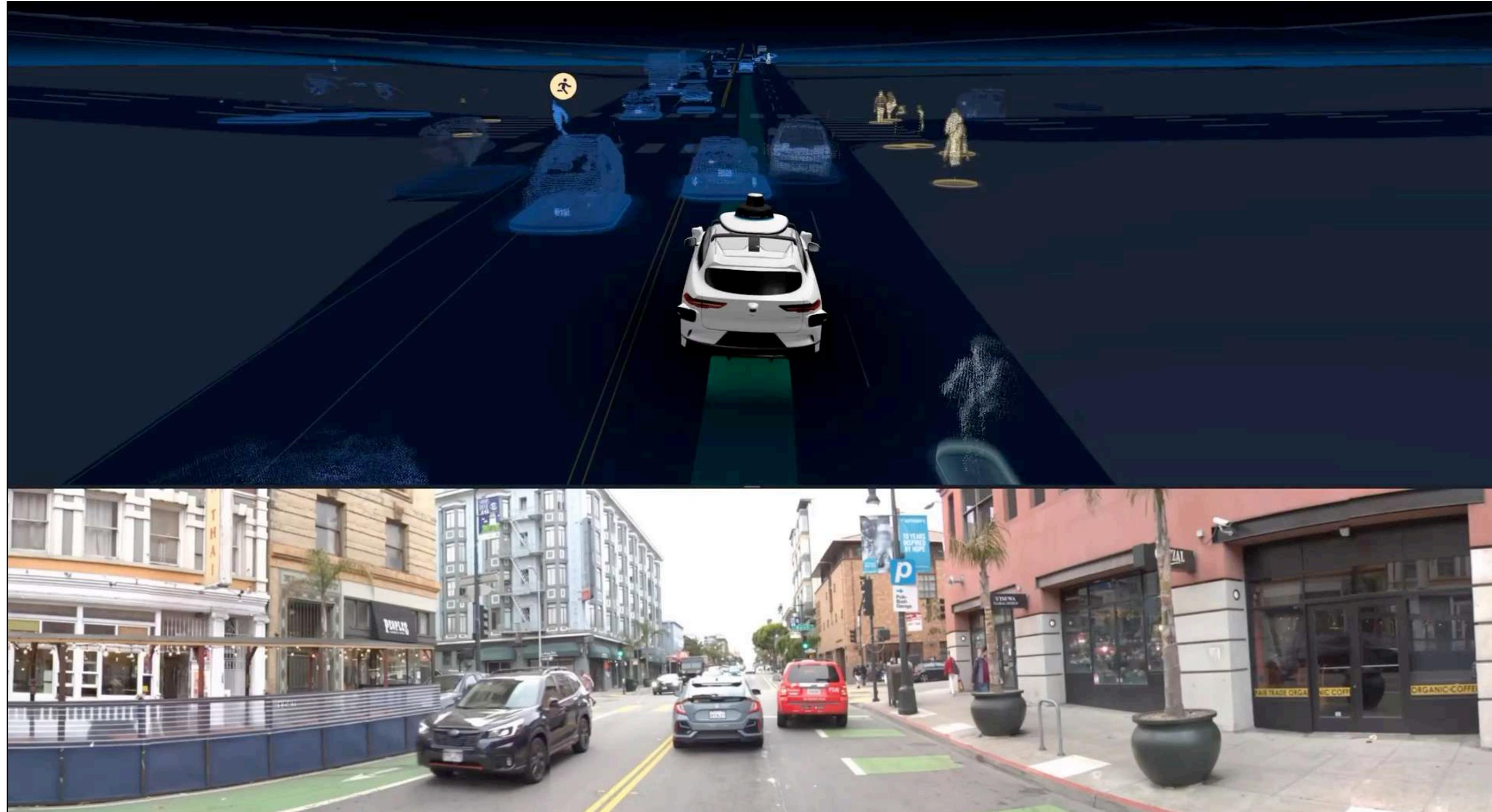


OpenAI Sora

Imagen Video: <https://openai.com/index/sora/>

From 2D Vision to 3D Vision

Deep learning helps machine perceive the surrounding environment



Waymo Driver

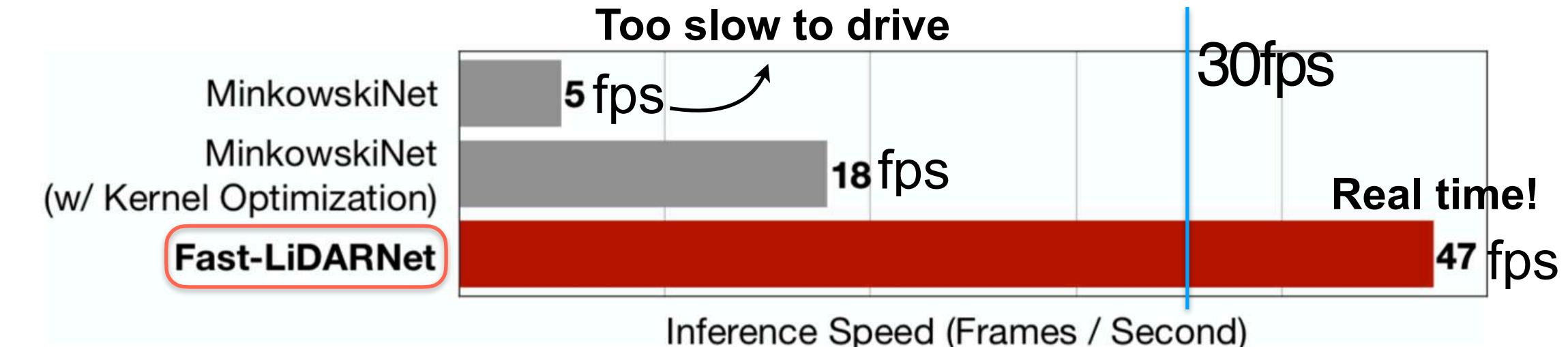
Waymo Driver: <https://www.youtube.com/watch?v=2CVInKMz9cA>



A **whole trunk** of workstation

Efficient 3D Perception

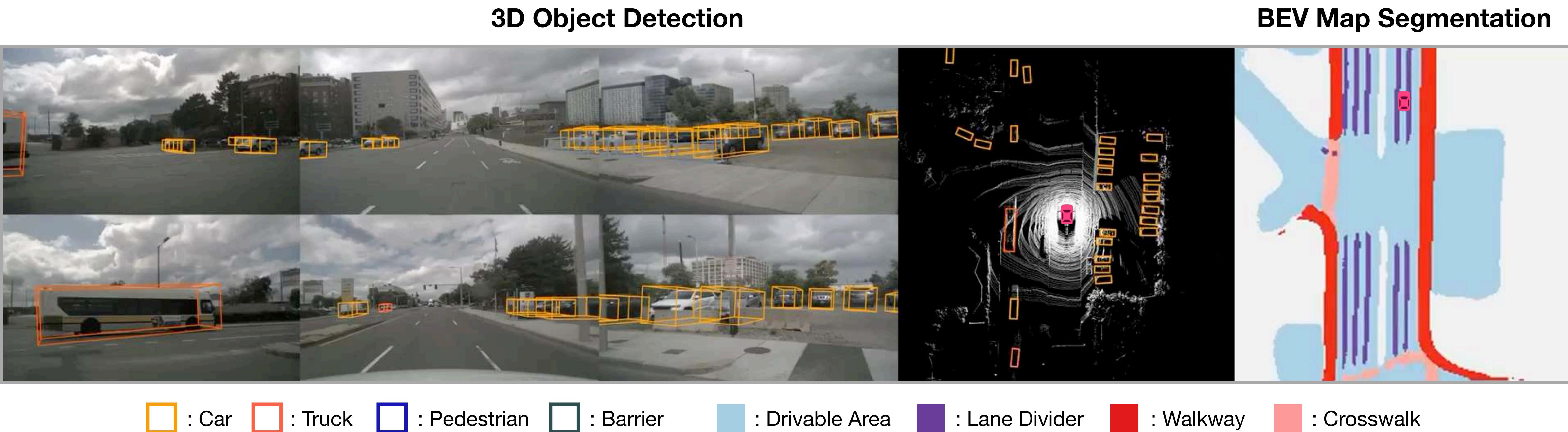
Fast-LiDARNet accelerates 3D perception with algorithm/system co-design



Efficient and Robust LiDAR-Based End-to-End Navigation [Liu et al., ICRA 2021]

Efficient 3D Perception

BEVFusion supports efficient multi-task multi-sensor fusion



BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation [Liu et al., Arxiv 2022]

Deep Learning is Everywhere



Image source: [1](#), [2](#), [3](#)

ChatGPT and Large Language Model (LLM)

LLMs are computationally heavy

HA Please draft a short description for the course "TinyML and Efficient Deep Learning Computing"?

 Course Title: "TinyML and Efficient Deep Learning Computing"

Course Description:

Unleash the power of Tiny Machine Learning (TinyML) and efficient deep learning computing in this comprehensive course designed for learners with a fundamental understanding of artificial intelligence (AI) and machine learning (ML).

Throughout this course, participants will gain a solid grasp of the emerging field of TinyML, which focuses on creating and implementing efficient machine learning models on resource-constrained devices such as microcontrollers. Our syllabus explores deep learning model optimization techniques, effective neural network architectures, and hardware-aware training approaches to make the most of limited computational capabilities.

Learners will explore real-world case studies and hands-on projects involving smartphones, IoT devices, and wearable technology, demonstrating the immense potential of TinyML applications in various sectors. By the end of the course, students will be equipped with the skills to design, develop, and deploy efficient deep learning models for edge computing, thus opening new opportunities for AI innovations that are cost-effective, power-efficient, and privacy-preserving.

We're experiencing exceptionally high demand. Please hang tight as we work on scaling our systems. 

ChatGPT is at capacity right now

[Get notified when we're back](#)

Our most capable model, great for tasks that require creativity and advanced reasoning.

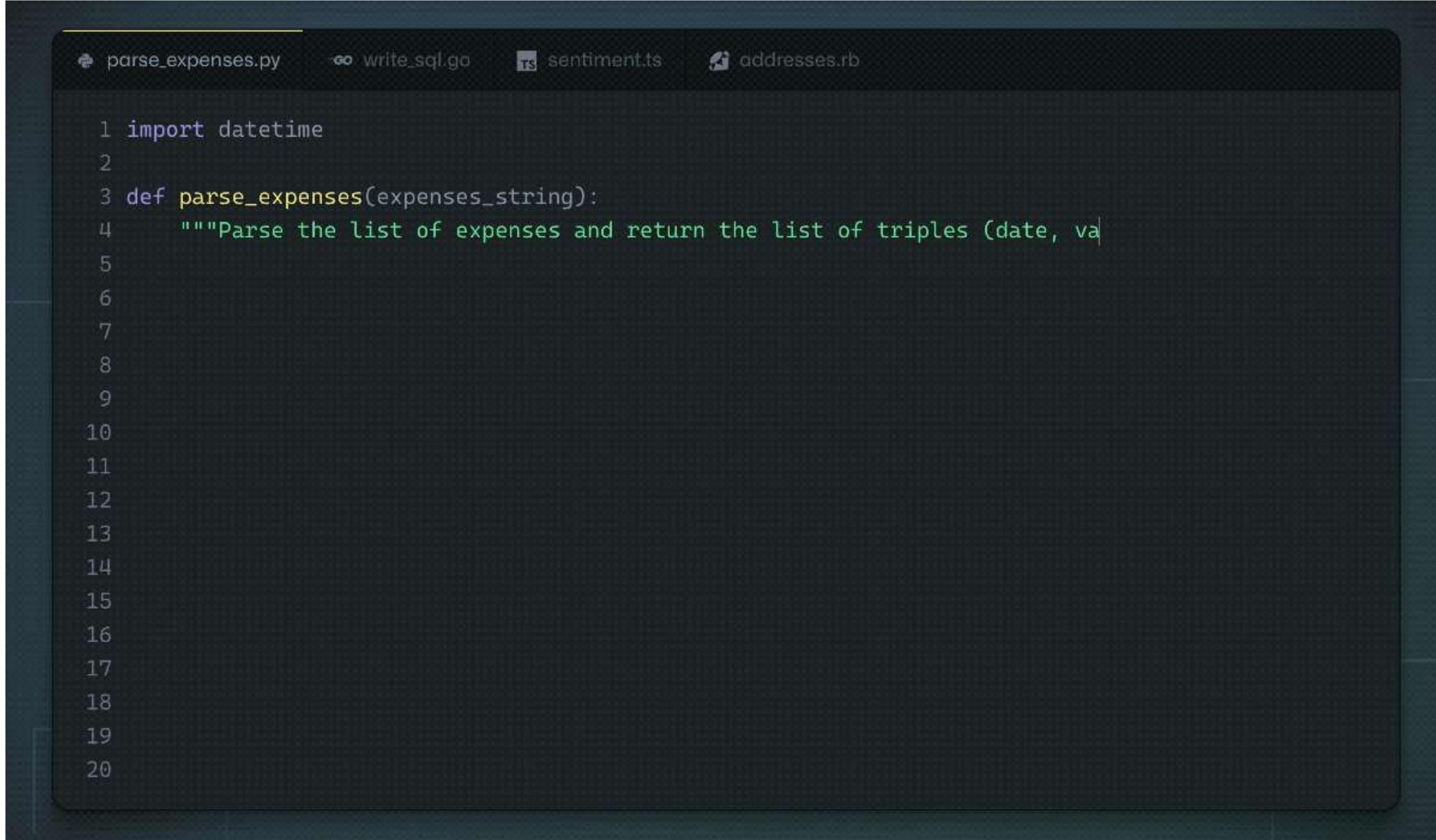
Available exclusively to Plus users

GPT-4 currently has a cap of 50 messages every 3 hours.

ChatGPT 

Code Generation

GitHub CoPilot can make meaningful coding suggestions based on context

A screenshot of a dark-themed code editor window. At the top, there are four tabs: 'parse_expenses.py' (active), 'write_sql.go', 'sentiment.ts', and 'addresses.rb'. The main area displays the following Python code:

```
1 import datetime
2
3 def parse_expenses(expenses_string):
4     """Parse the list of expenses and return the list of triples (date, va
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

The code starts with an import statement for 'datetime'. It then defines a function 'parse_expenses' that takes a parameter 'expenses_string'. A multi-line docstring begins with '"""Parse the list of expenses and return the list of triples (date, va'. The code ends with a closing parenthesis for the function definition and a final closing brace for the class definition.

Image credit: <https://techcrunch.com/2021/06/29/github-previews-new-ai-tool-that-makes-coding-suggestions/>

Neural Machine Translation

Neural machine translation bridges the language barrier

The screenshot shows the Google Translate mobile application. At the top, there are language selection bars for English (DETECTED), English, Spanish, French, and Chinese (Simplified). Below these, there are two text boxes. The left text box contains an English paragraph about deploying neural networks on mobile devices. The right text box displays the Chinese translation of this paragraph. At the bottom of the right text box, there is a link to "Show more".

ENGLISH - DETECTED ENGLISH SPANISH FRENCH CHINESE (SIMPLIFIED) ENGLISH SPANISH

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, 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.

您是否发现很难在移动设备和物联网设备上部署神经网络？你有没有发现训练神经网络太慢了？本课程深入探讨有效的机器学习技术，这些技术可在资源受限的设备上实现强大的深度学习应用。主题涵盖高效推理技术，包括模型压缩、剪枝、量化、神经架构搜索、蒸馏；和高效的训练技术，包括梯度压缩和设备迁移学习；其次是针对视频、点云和 NLP 的特定应用模型优化技术；和高效的量子机器学习。学生将通过与移动 AI 相关的开放式设计项目获得在微控制器、手机和量子机器上实施深度学习应用程序的实践经验。

Nín shìfǒu fāxiàne hěn nán zài yídòng shèbèi hé wù liánwǎng shèbèi shàng bùshǔ shénjīng wǎngluò? Nǐ yǒu méiyǒu fāxiàne xùnliàn shénjīng wǎngluò tài mǎnle? Běn kèchéng shēnrù tantǎo yóuxiào de jīqì xuéxí jishù, zhèxiē jishù kě zài zīyuán shòu xiàn de

Show more

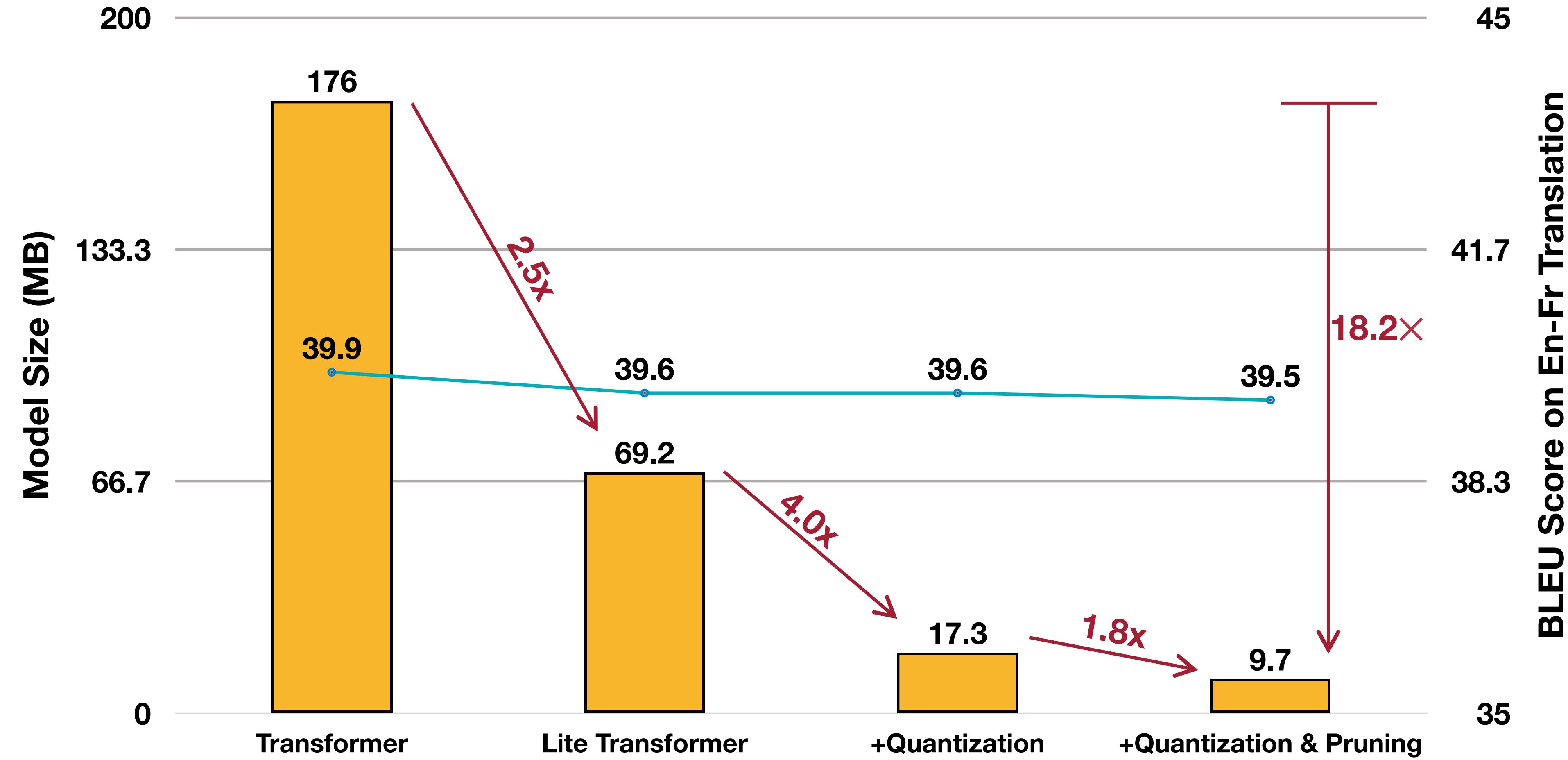


Conversation Translation
(on iPhone)

Google Translate: <https://translate.google.com/>

Efficient Neural Machine Translation

Lite Transformer reduces the model size with pruning and quantization



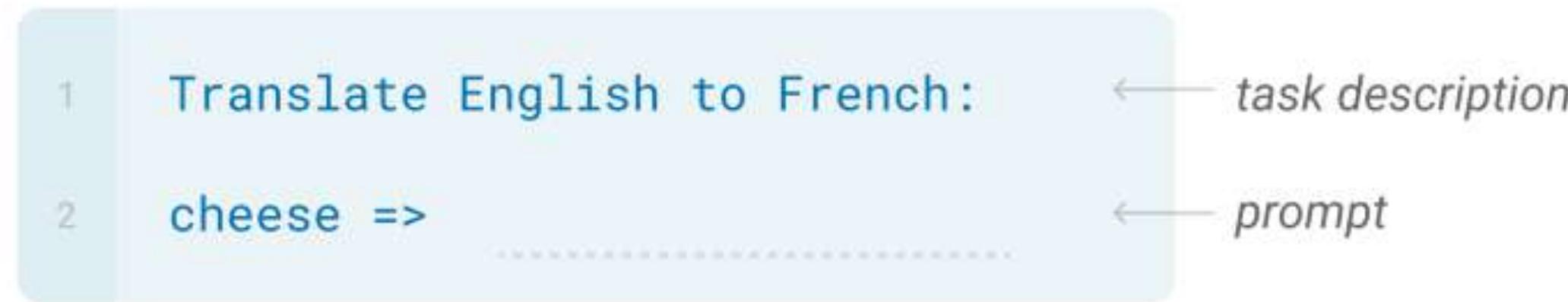
Lite Transformer with Long-Short Range Attention [Wu et al., ICLR 2020]

Large Language Models

Large language models show emergent behaviors: zero/few learning

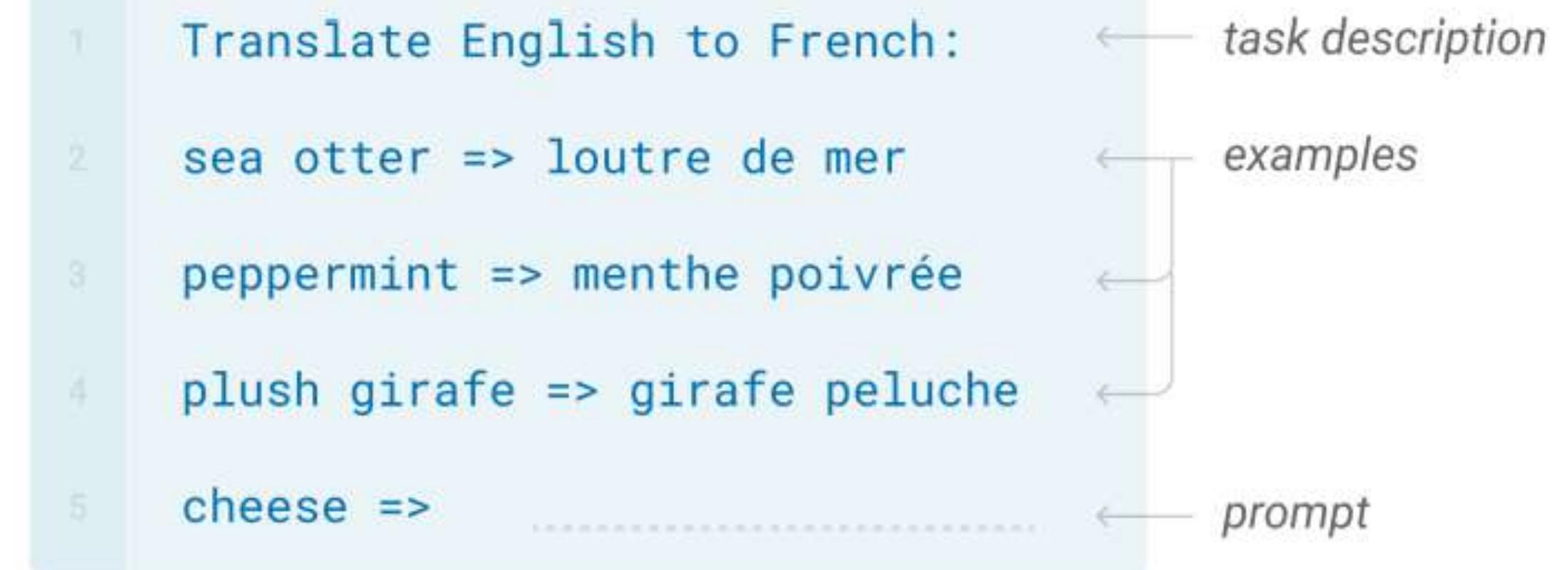
Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



Few-shot

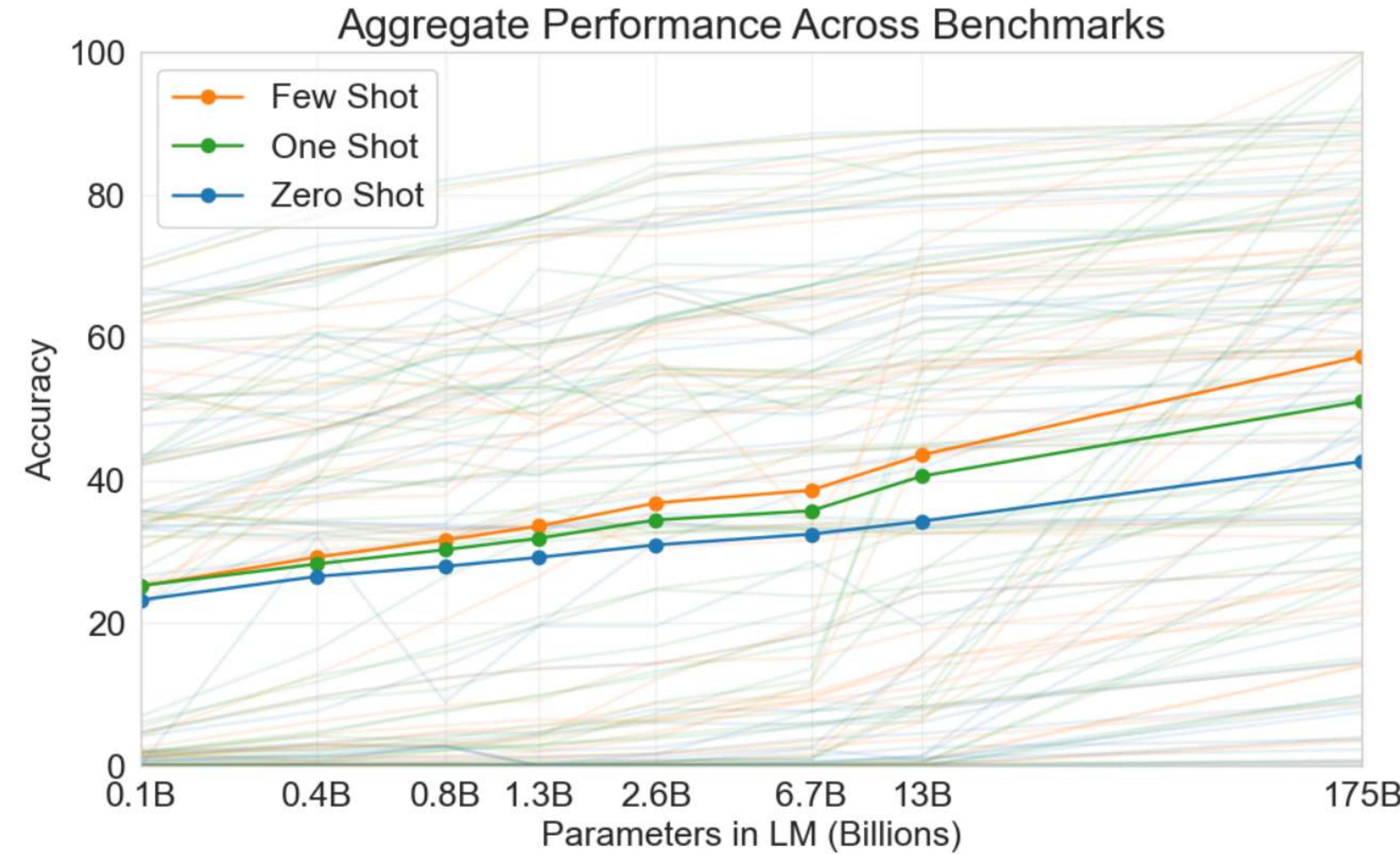
In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



Brown et al., GPT-3, 2020

Large Language Models

But it comes at the cost of large model size



Brown et al., GPT-3, 2020

Large Language Models

Large language models show emergent behaviors: chain-of-thought

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. 

Large Language Models

Large language models show emergent behaviors: chain-of-thought

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. 

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

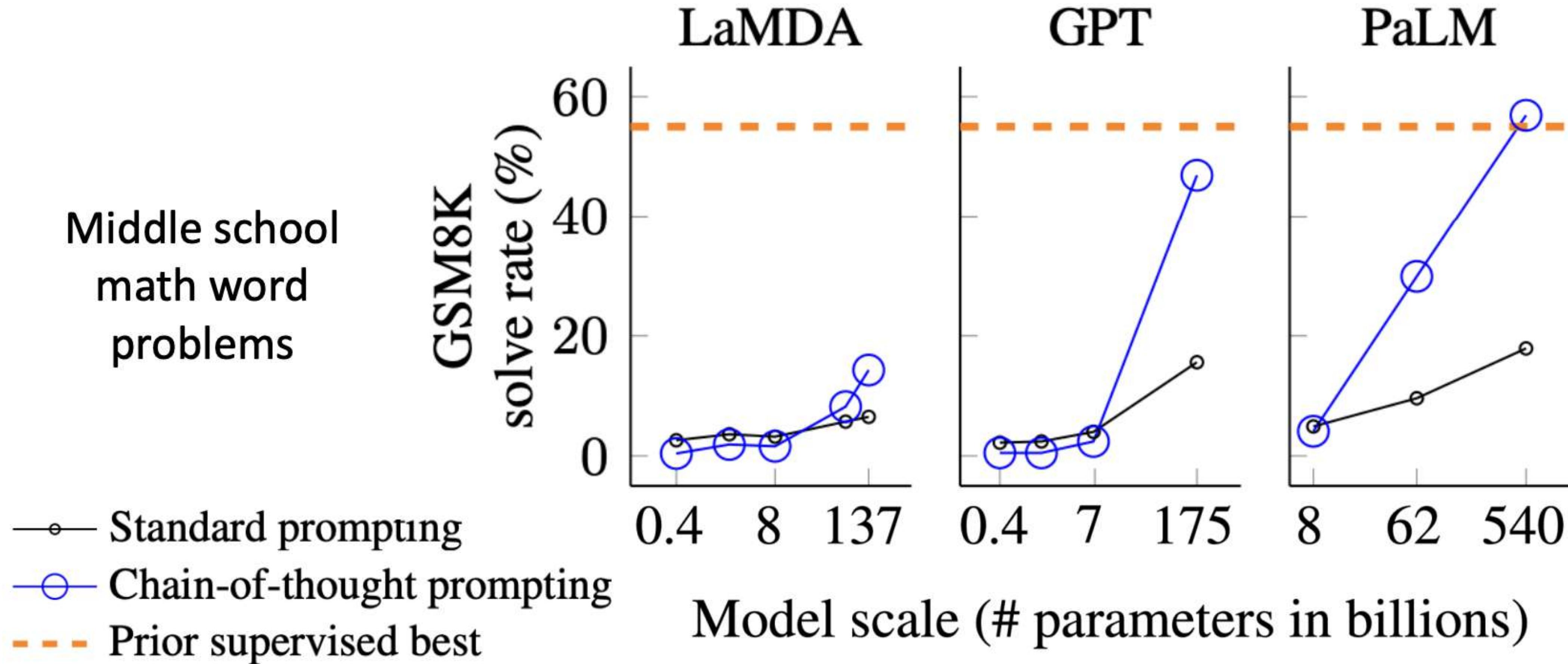
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. 

Large Language Models

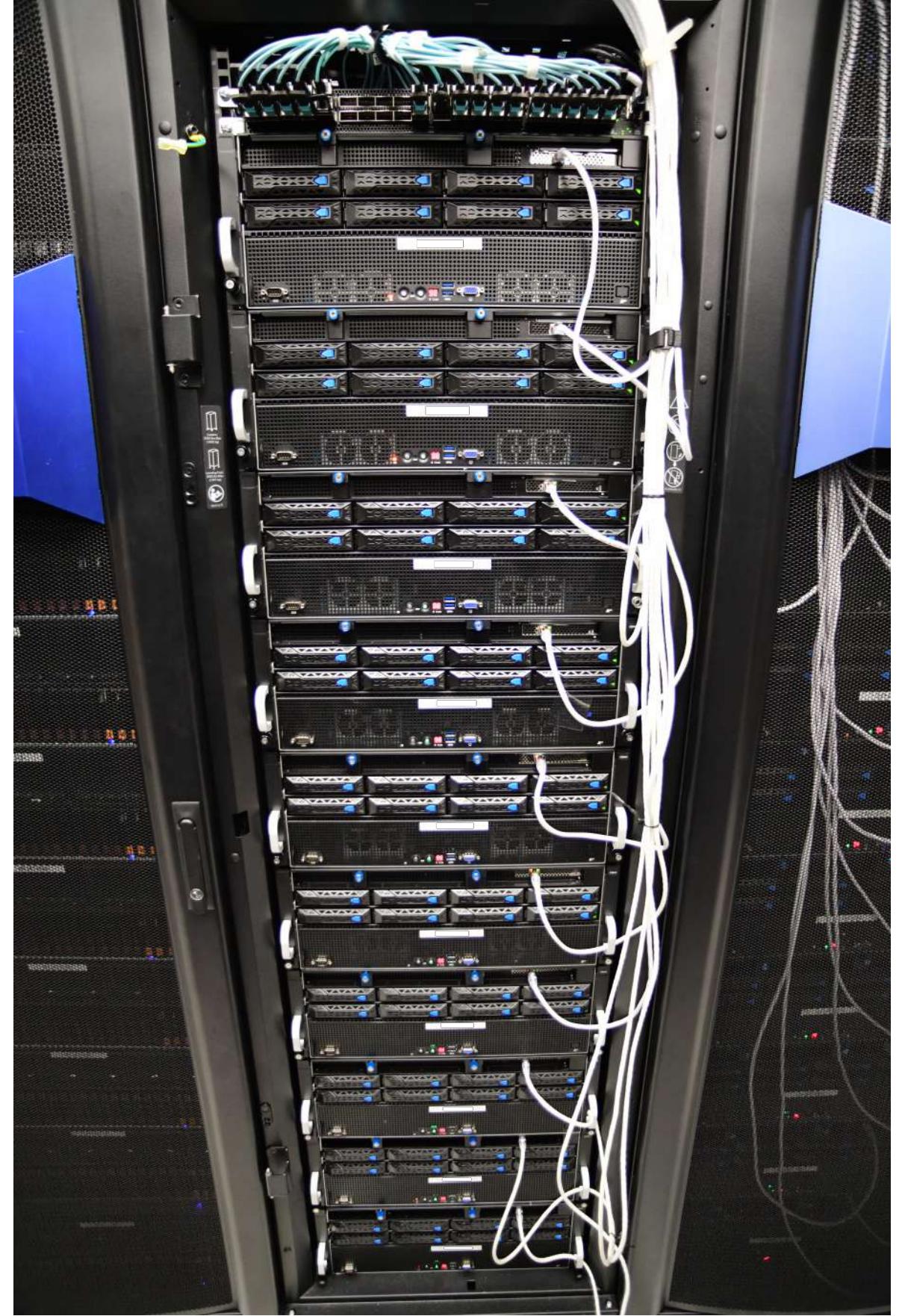
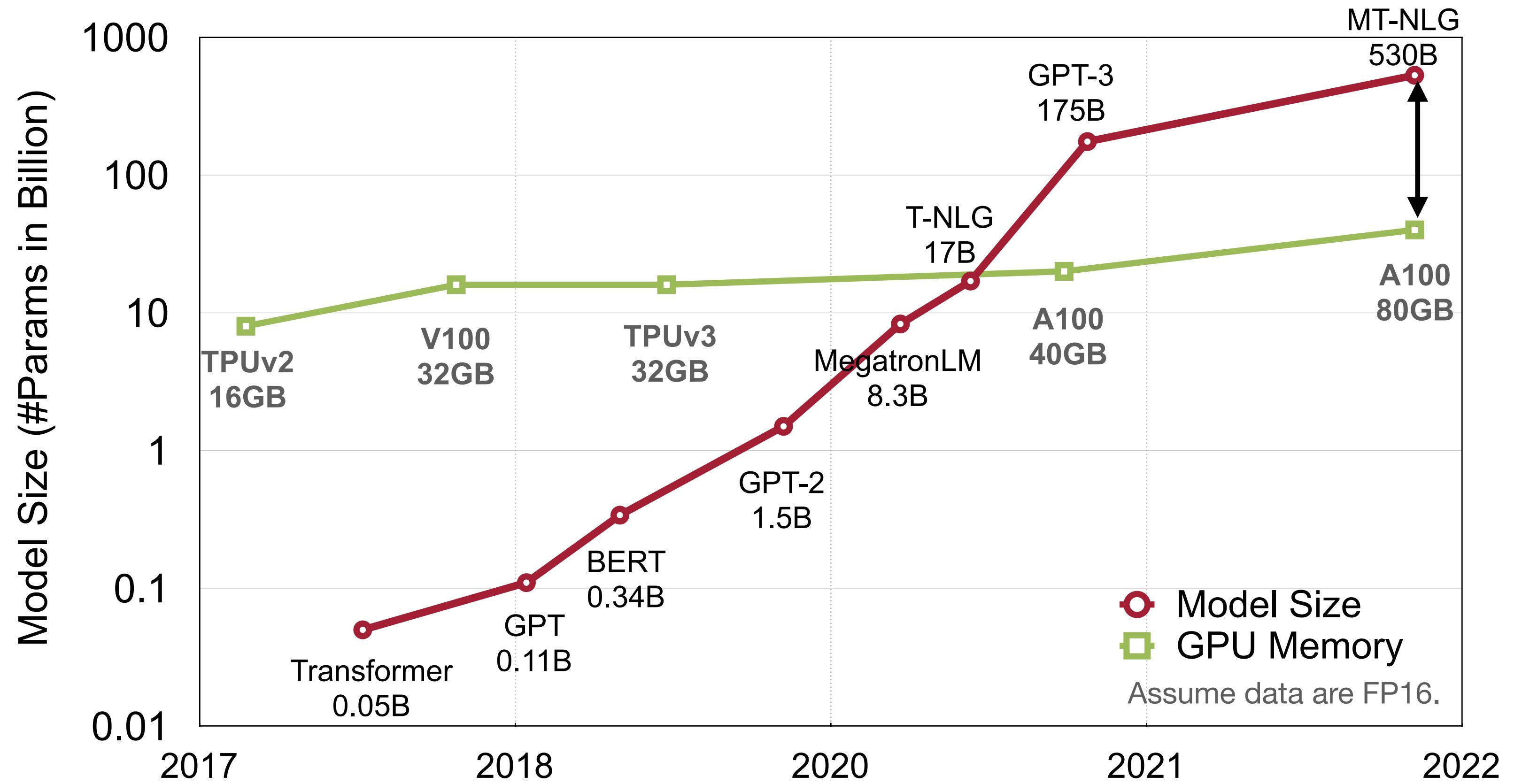
But it comes at the cost of large model size



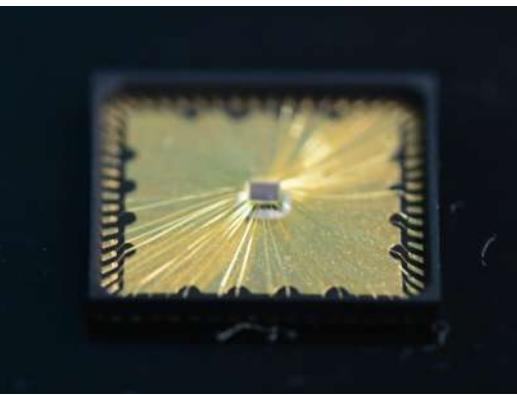
Wei et al., Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, 2022

Large Language Models

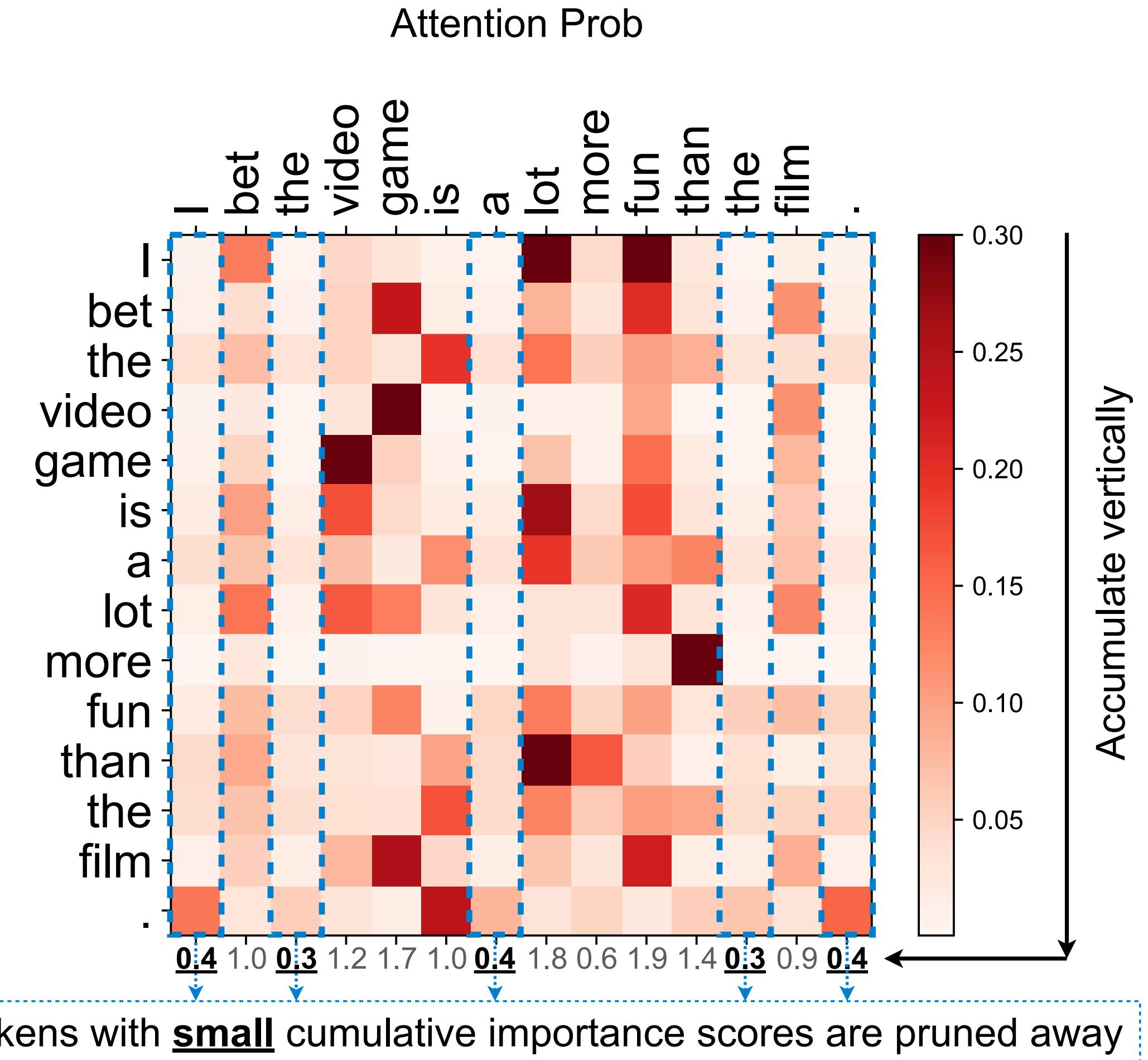
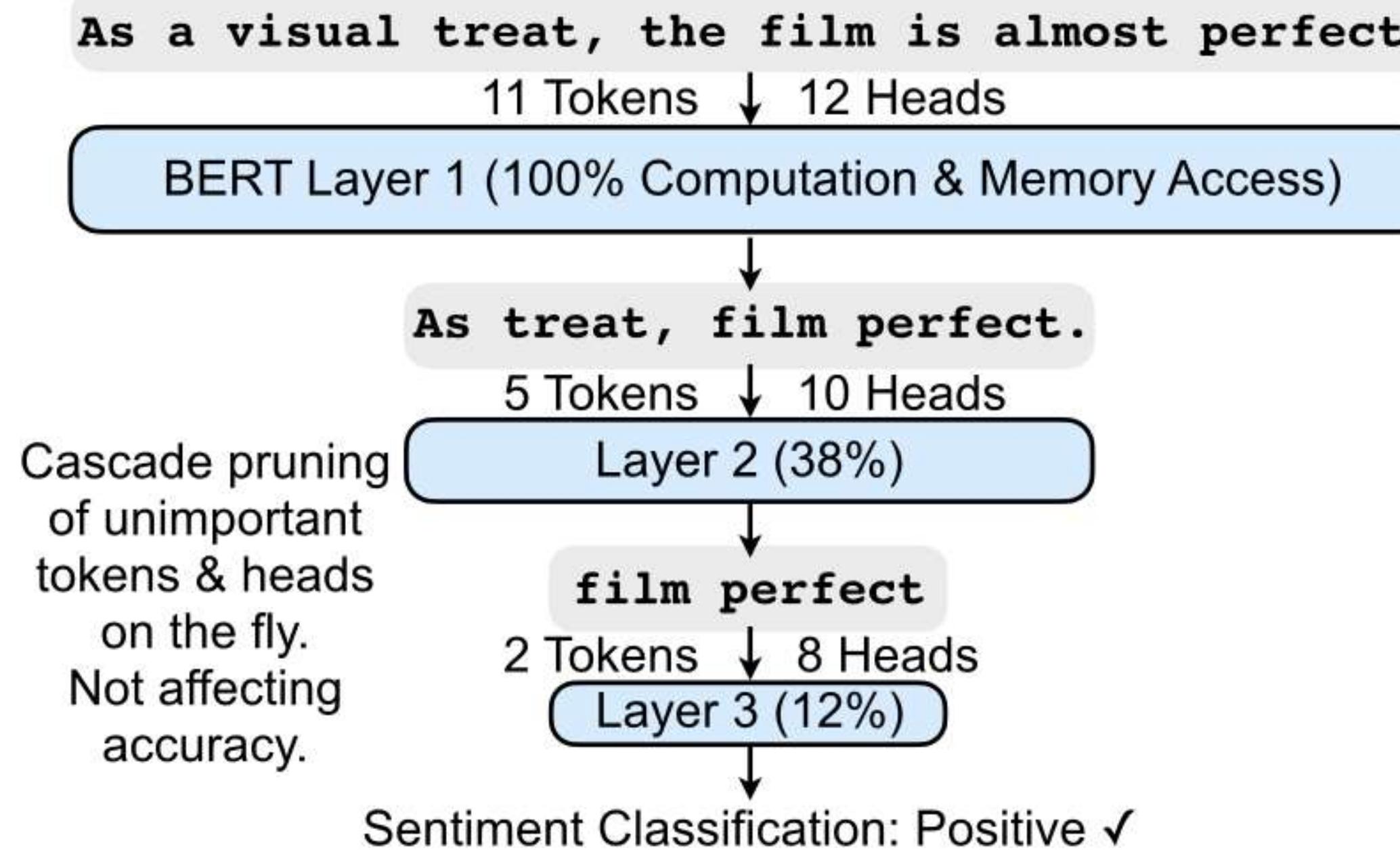
Model size of language models is growing fast



Efficient Large Language Models

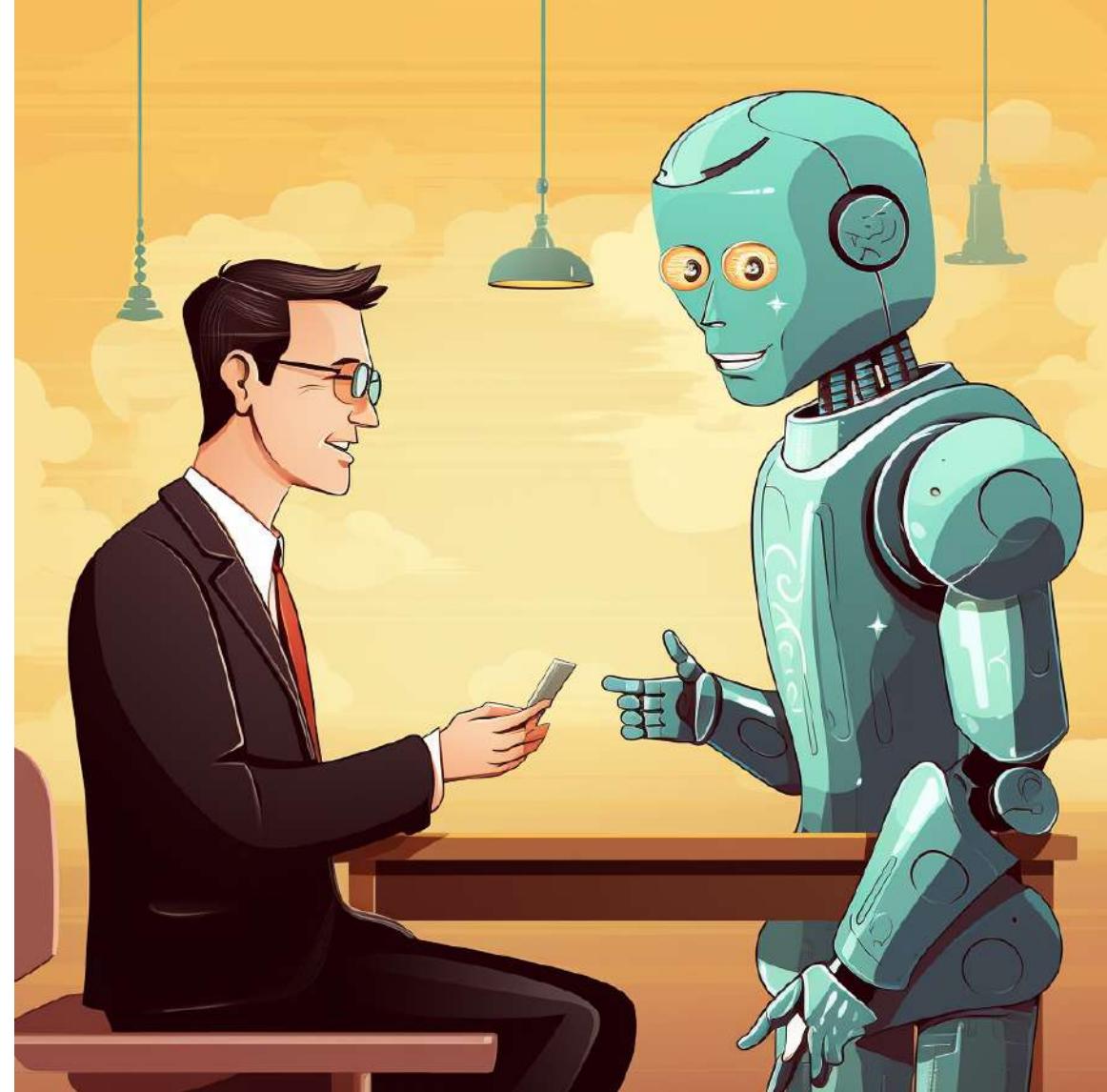
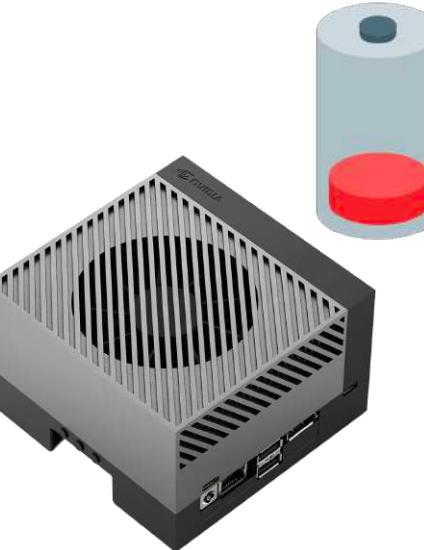


SpAtten accelerates language models by pruning redundant tokens



Efficient Large Language Models

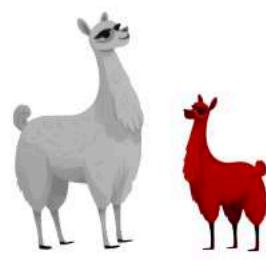
Running LLMs on the edge is very important



- Deploying LLM on the edge is useful: running copilot services (code completion, office, game chat) locally on laptops, cars, robots, and more. These devices are **resource-constrained**, **low-power** and sometimes **do not have access to the Internet**.
- **Data privacy** is important. Users do not want to upload personal data to the cloud.

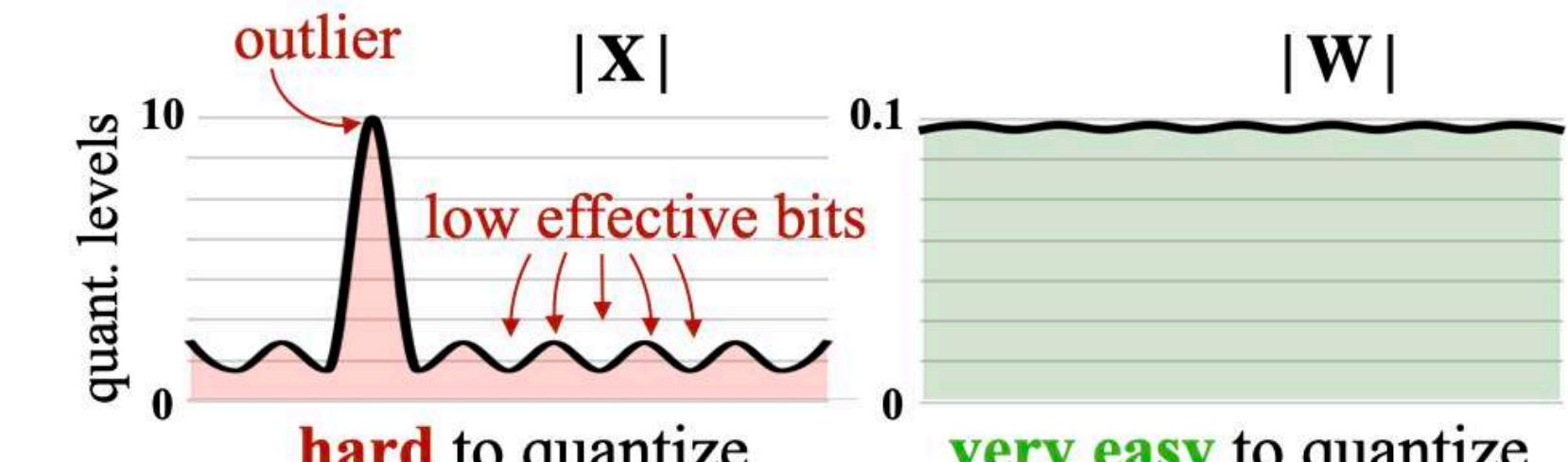
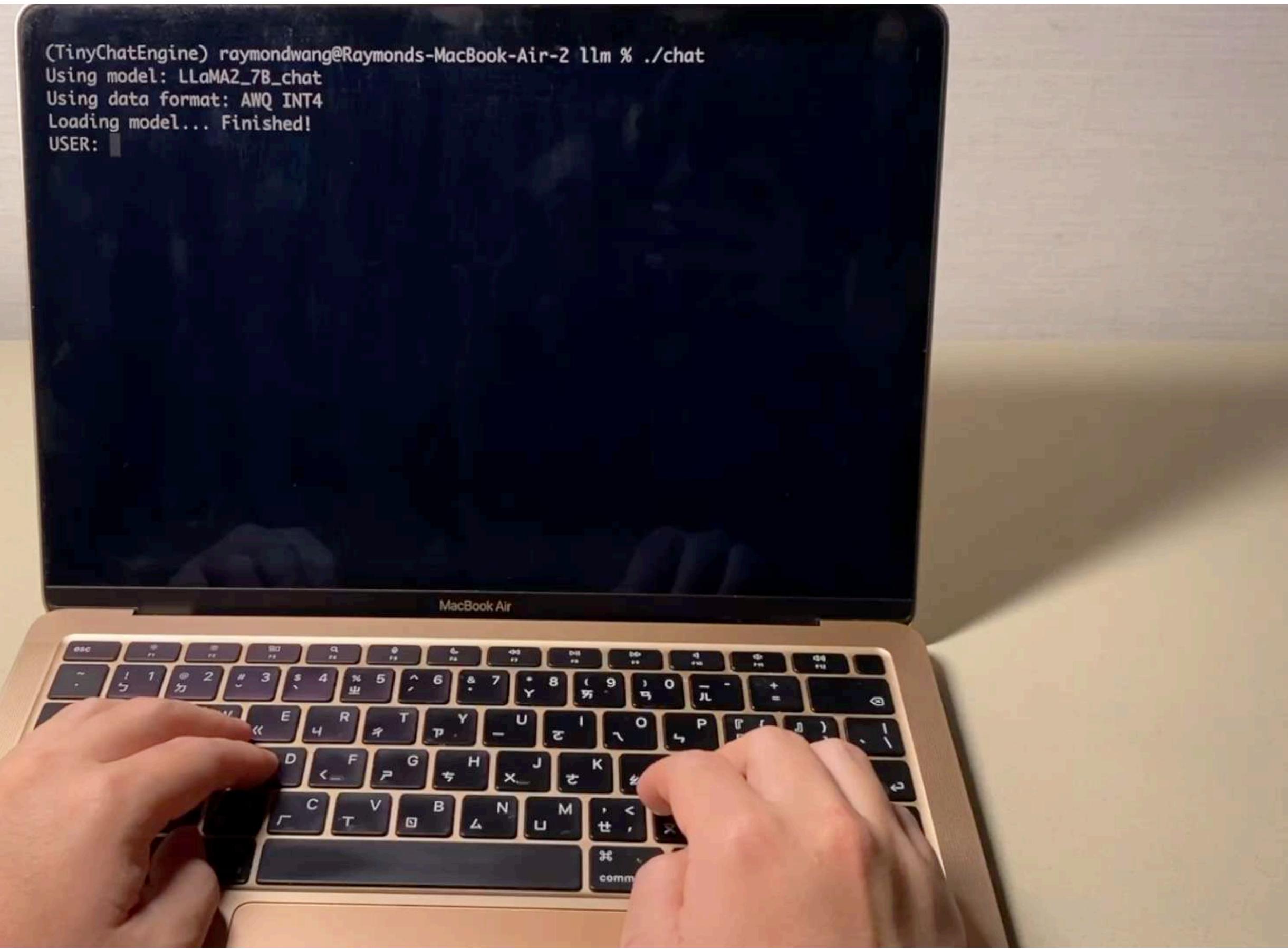
Images are generated by Midjourney

LLM Quantization

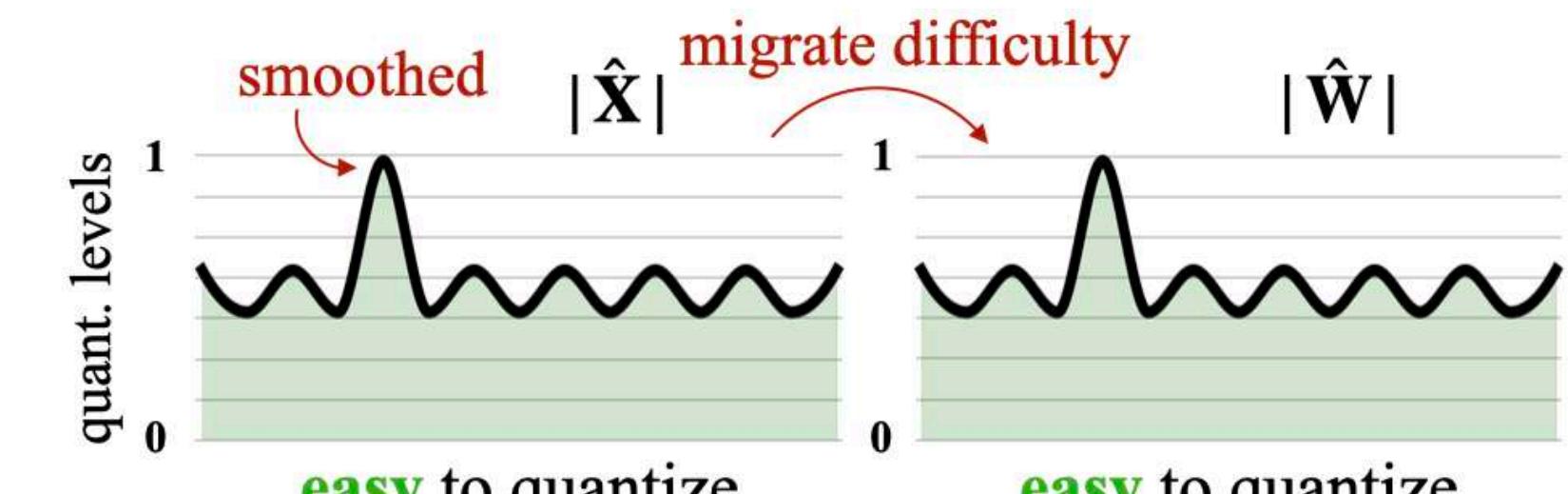


TinyChat

- LLMs are too large to fit into the memory of edge devices. We enable edge deployment of LLMs through quantization: SmoothQuant and AWQ
- TinyChatEngine implements the compressed inference, built from C/C++ from scratch, easy to install and migrate to edge platforms



(a) Original



(b) SmoothQuant (ours)

TinyChatEngine: <https://github.com/mit-han-lab/TinyChatEngine>

TinyChat: a lightweight LLM inference engine

Demo on TinyChatComputer, powered by NVIDIA Jetson Orin Nano

- On a GPU board with just ~7G available memory, TinyChat enables efficient deployment of 7B large language models, thanks to AWQ quantization.
- Importance of edge LLM deployment: **data privacy and security, continued operation in disconnected environment, customization and personalization.**



TinyChat on Orin

Running LLMs on resource-constrained edge GPUs



TinyChat on Orin

TinyChat runs LLM on the edge



```
(AWQ-Chat) orin% ./llama2_awq_int4.sh 7b
Loading checkpoint: 100%
USER: Describe five attractions in the Greater Boston Area.
```

LLaMA-2-7B (W4A16, AWQ): 30 tokens / s

TinyChat on Orin

TinyChat enables local inference of 13B LLMs on edge devices



A screenshot of a Visual Studio Code interface running on an Orin device via SSH. The terminal tab is active, showing the command `./llama2_awq_int4.sh 13b` being run. The output indicates a 100% checkpoint load and a user query: "USER: How do you compare MIT and Harvard?". The assistant's response is shown as a short vertical bar. The interface includes a sidebar with icons for problems, output, ports, and debug console, and a bottom status bar showing the connection details.

LLaMA-2-13B (W4A16, AWQ): 17 tokens / s

AWQ 4-bit Quantization for LLM



AWQ brings about 3.3x speedup to LLaMA-2 on 4090

shang [SSH: Hanlab_4090] - Visual Studio Code

TERMINAL

```
(AWQ-Chat) x4% ./llama2_fp16.sh 7b
Loading checkpoint shards: 100%|██████████| 2/2 [00:05<00:00,  2.70s/it]
USER:
```

shang [SSH: Hanlab_4090] - Visual Studio Code

TERMINAL

```
(AWQ-Chat) x4% ./llama2_awq_int4.sh 7b
Loading checkpoint: 100%|██████████| 1/1 [00:01<00:00,  1.27s/it]
USER:
```

LLaMA-2-7B (FP16): 50 tokens / s



LLaMA-2-7B (W4A16, AWQ): 166 tokens / s*

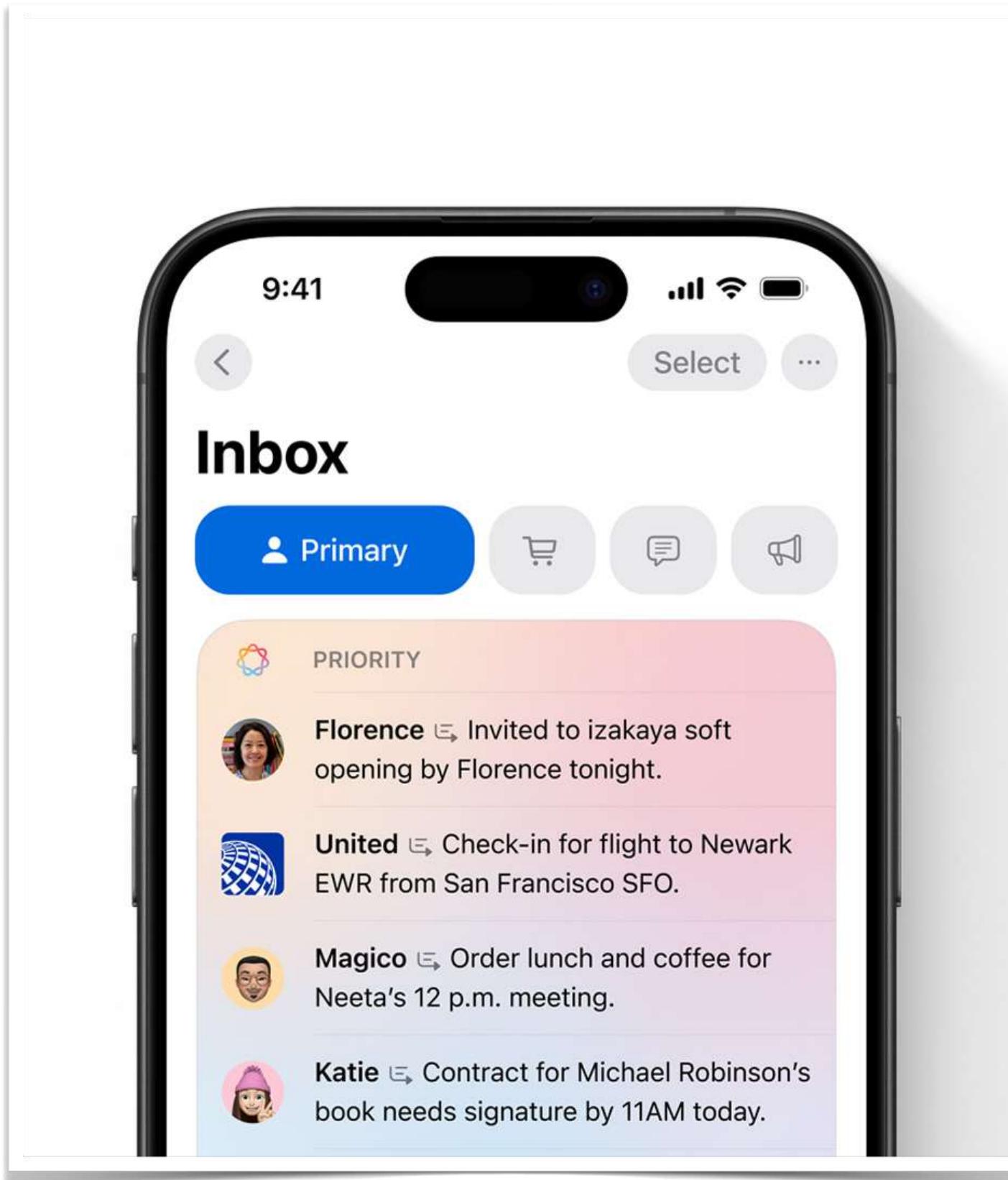
All-Python user interface, diverse and flexible support for different LLMs (e.g. LLaMA, MPT, Falcon)

*: If measured in exLLaMA's setting, we can achieve 195 tokens / s.

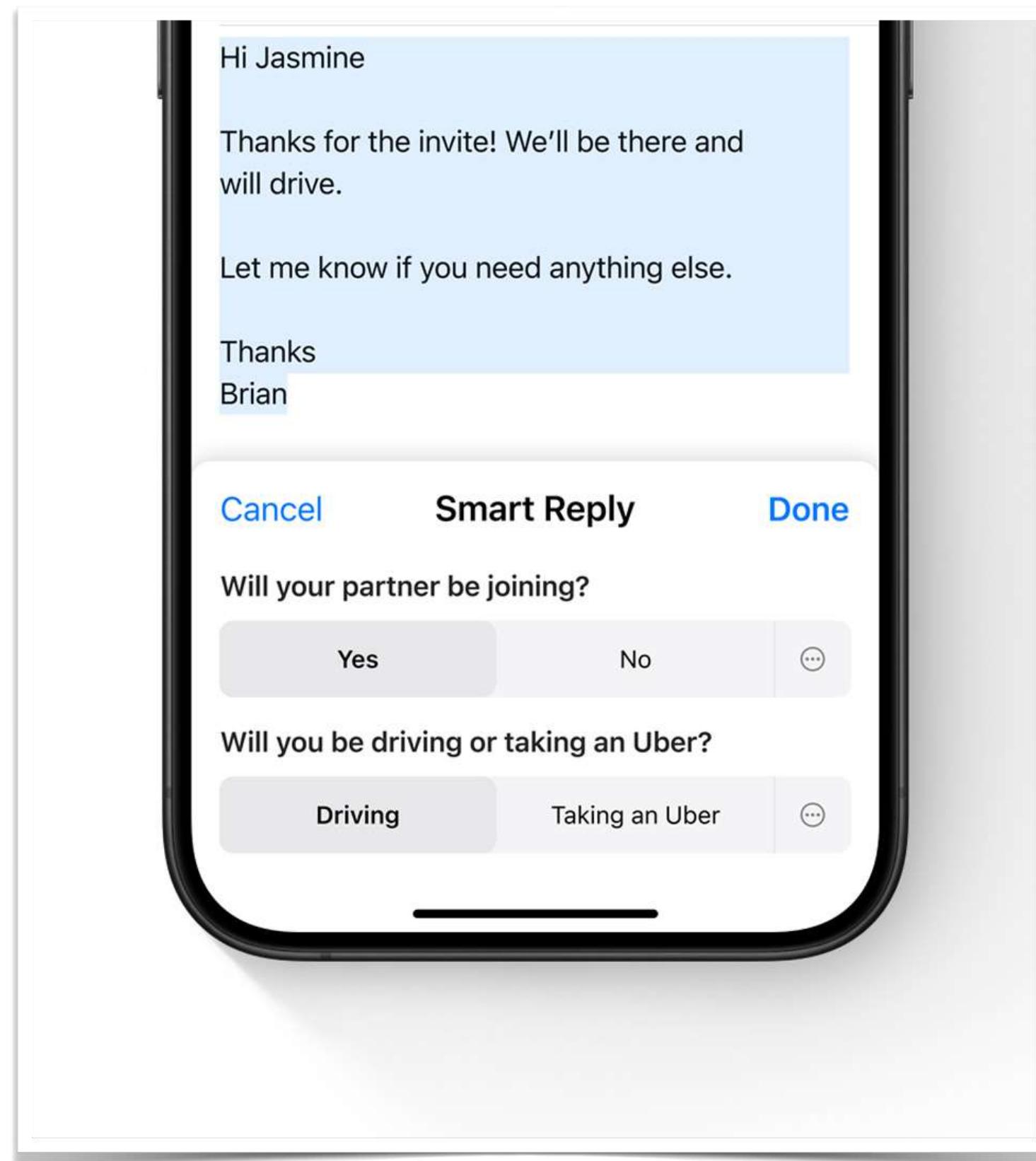
TinyChat: <https://tinychat.hanlab.ai>

LLM on Phones

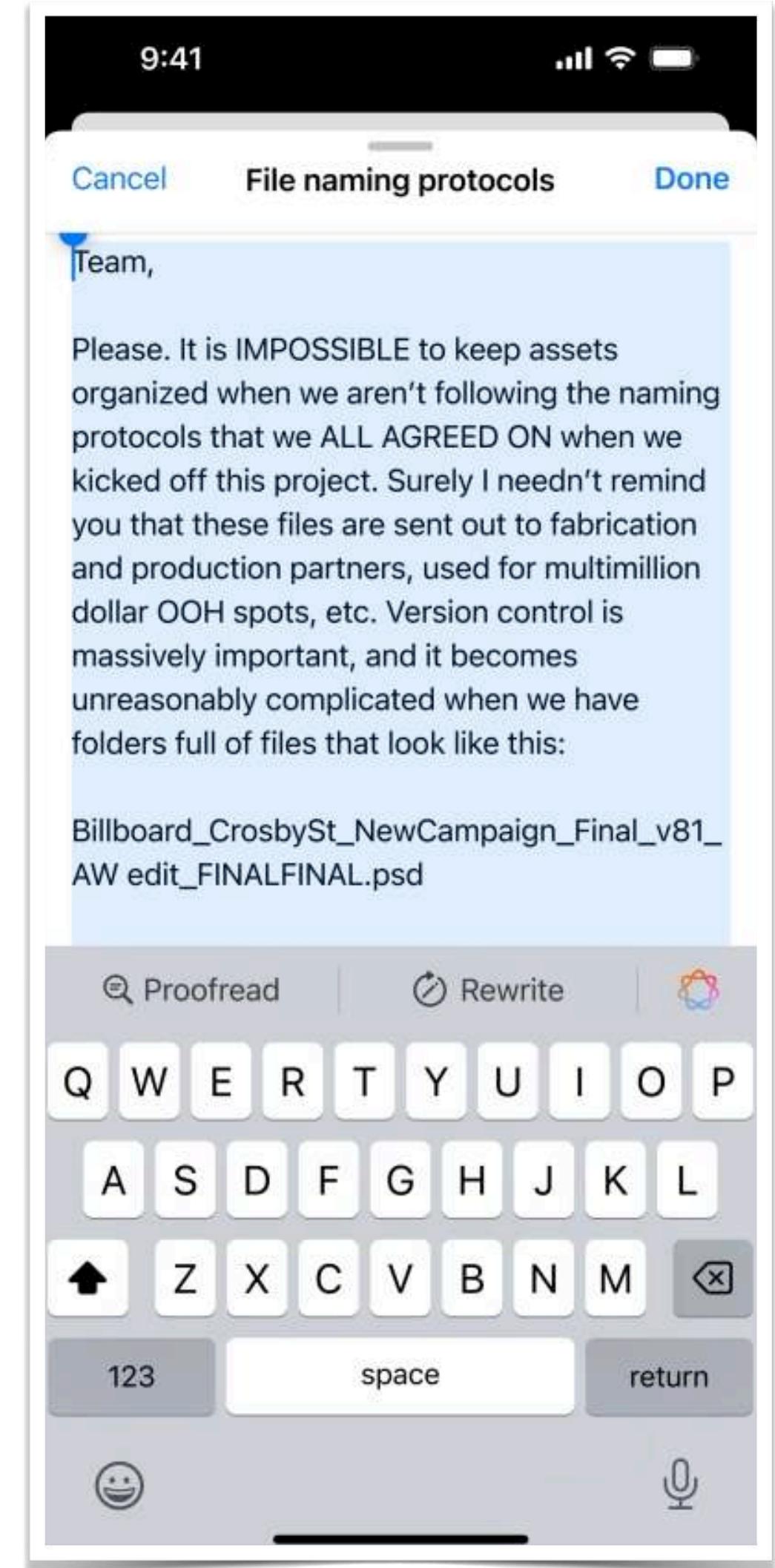
LLM empowers writing tools on phones.



Summarization



Smart Reply



Writing Tools

Source: <https://www.apple.com/apple-intelligence/>

Deep Learning is Everywhere



Image source: [1](#), [2](#), [3](#)

Vision-Language Models

LLaVA achieves general-purpose visual and language understanding

The screenshot shows a conversation in a messaging interface. The user uploads an image of the Mona Lisa painting. The user then asks, "Do you know who drew this painting?". The LLaVA model responds with a detailed description: "The painting depicts a woman, commonly believed to be Mona Lisa, the famous artwork by Leonardo da Vinci. It is a portrait painting that showcases the woman's enigmatic smile and has become one of the most famous and iconic art pieces in the world. The original work is displayed in the Louvre Museum in Paris, and it is known for its intricate details, use of oil paint, and the artist's innovative techniques that contributed to its enduring appeal and mystery."

LLaVA uses a 13B LLaMA for language understanding.

LLaVA: <https://llava-vl.github.io/>



Efficient Vision-Language Models

AWQ quantizes vision-language models to 4 bits with high quality



Question: Do you know who drew this painting?

INT4-g128 RTN: The painting depicts a woman, who some believe to be a representation of the famous artist, Mona Lisa. However, it is not certain that the woman in the painting is indeed Mona Lisa. The artwork is a popular subject of artistic analysis and admiration, often considered a symbol of feminine beauty and enigma. The portrait is characterized by its intricate details, captivating the viewer with the woman's expressive gaze and the subtle, enigmatic smile that has become synonymous with the Mona Lisa.

INT4-g128 AWQ: The painting depicts a woman, who is believed to be a portrait of Mona Lisa, the famous artwork by the artist **Leonardo da Vinci**. Although the painting is widely recognized as a representation of Mona Lisa, it is not possible to confirm the artist who drew this particular painting without additional information. Leonardo da Vinci was a renowned artist known for his exceptional skill in painting detailed, realistic portraits, and the Mona Lisa is one of his most famous works.

Sometimes I just look at pictures of the earth from space and I marvel at how beautiful it all is.

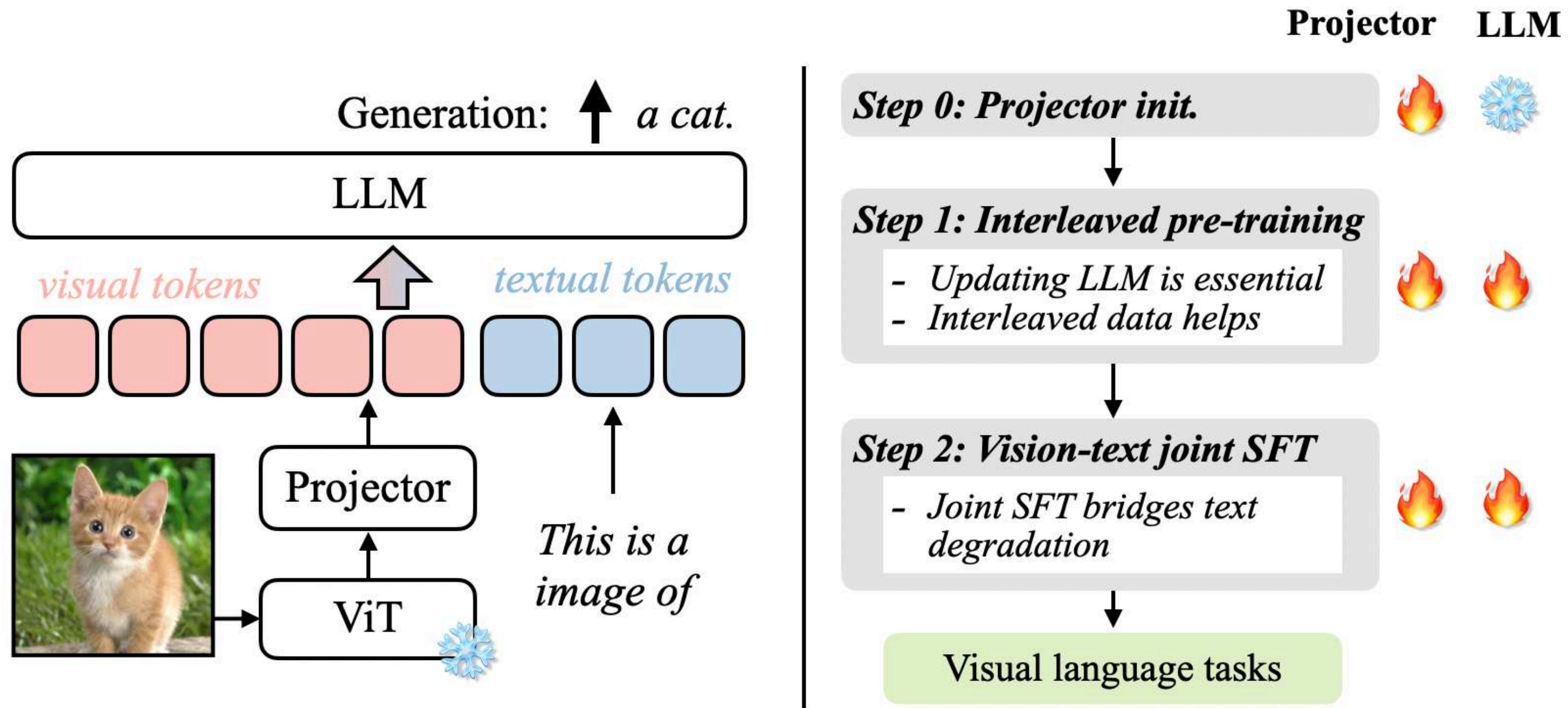


Question: Can you explain this meme in detail?

INT4-g128 RTN: The meme in the image features a plate of food, likely fried chicken, with a unique twist. **There are small pictures of the Earth and other planets placed on top of the food**, creating an interesting and amusing visual. Alongside the plate, a caption reads, "Sometimes I look at pictures of the Earth from space and I marvel at how beautiful it all is." This combination of food and planet images adds a touch of humor and curiosity, making the viewer wonder about the context and intent behind the meme.

INT4-g128 AWQ: The meme in the image is a **lighthearted and humorous take on the concept of looking at pictures of the Earth from space**. A plate of fried food, specifically chicken nuggets, is presented with the caption, "Sometimes I look at pictures of the Earth from space and I marvel at how beautiful it all is." The playful combination of a food item and an Earth-related context creates a fun and entertaining visual.

ViLA Visual Language Model



- Data and training recipe matters more than architecture.
- Visual instruction tuning is not enough; we need visual language pre-training.
- Image-text pairs are not enough; interleaved image text data is crucial.
- Visual in-context learning requires unfreezing the LLM.
- Re-blending text-only data recovers text-only capability

Efficient Vision-Language Models

VILA: On Pre-training for Visual Language Models



Prompt: Please tell me what happens in the video.

VILA: The video shows a soccer game where a player scores a goal. The crowd cheers as the player celebrates with his teammates.

<https://vila.mit.edu/>

Efficient Vision-Language Models

VILA: On Pre-training for Visual Language Models



Prompt: According to the video, what will likely happen next?

VILA: The video shows a blender with strawberries inside, and it is likely that the blender will be turned on to blend the strawberries into a smoothie.

Efficient Vision-Language Models

VILA: On Pre-training for Visual Language Models



Prompt: What will likely happen next?

VILA: The next step in the video is likely to involve the person grinding the spices in the mortar.

Efficient Vision-Language Models

VILA: On Pre-training for Visual Language Models



Prompt: <image> Boston. <image> Toronto. <image>
VILA: San Francisco.

<https://vila.mit.edu/>

Efficient Vision-Language Models

VILA: On Pre-training for Visual Language Models

Prompt: Where is this?



<https://vila.mit.edu/>

Efficient Vision-Language Models

VILA: On Pre-training for Visual Language Models



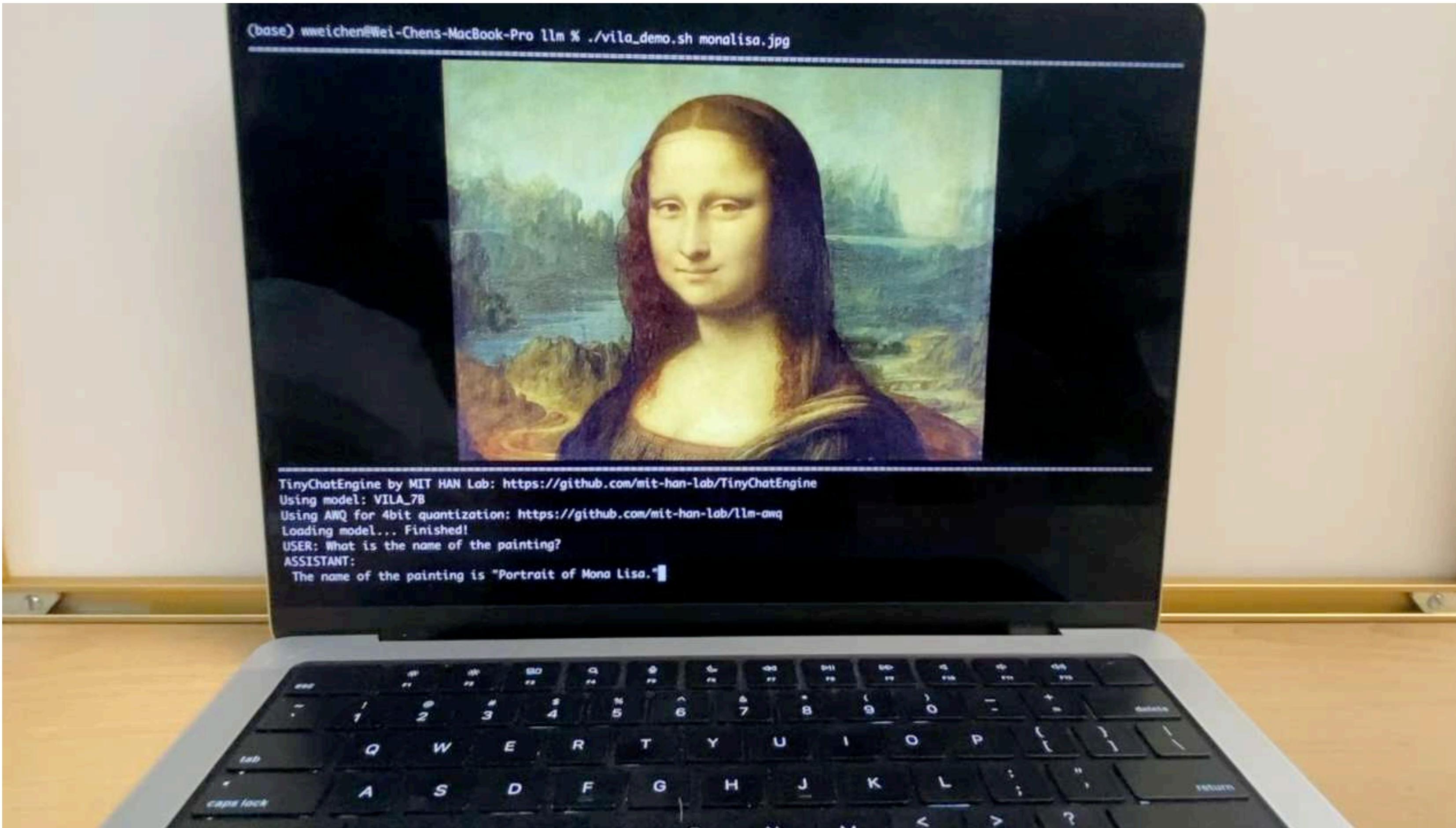
Prompt: Where is this?

VILA: This is the ancient city of Pompeii, located in Italy. The image captures the ruins of the city, with a statue standing tall amidst the remnants of the past. The statue, a symbol of resilience, overlooks the city, silently narrating tales of a time long gone. The ruins, a testament to the city's rich history, stand as a reminder of the passage of time. The image beautifully encapsulates the blend of history and nature, painting a picture of a city that once was, now forever preserved in stone.

<https://vila.mit.edu/>

ViLA On-Device

Run visual language models on MacBook

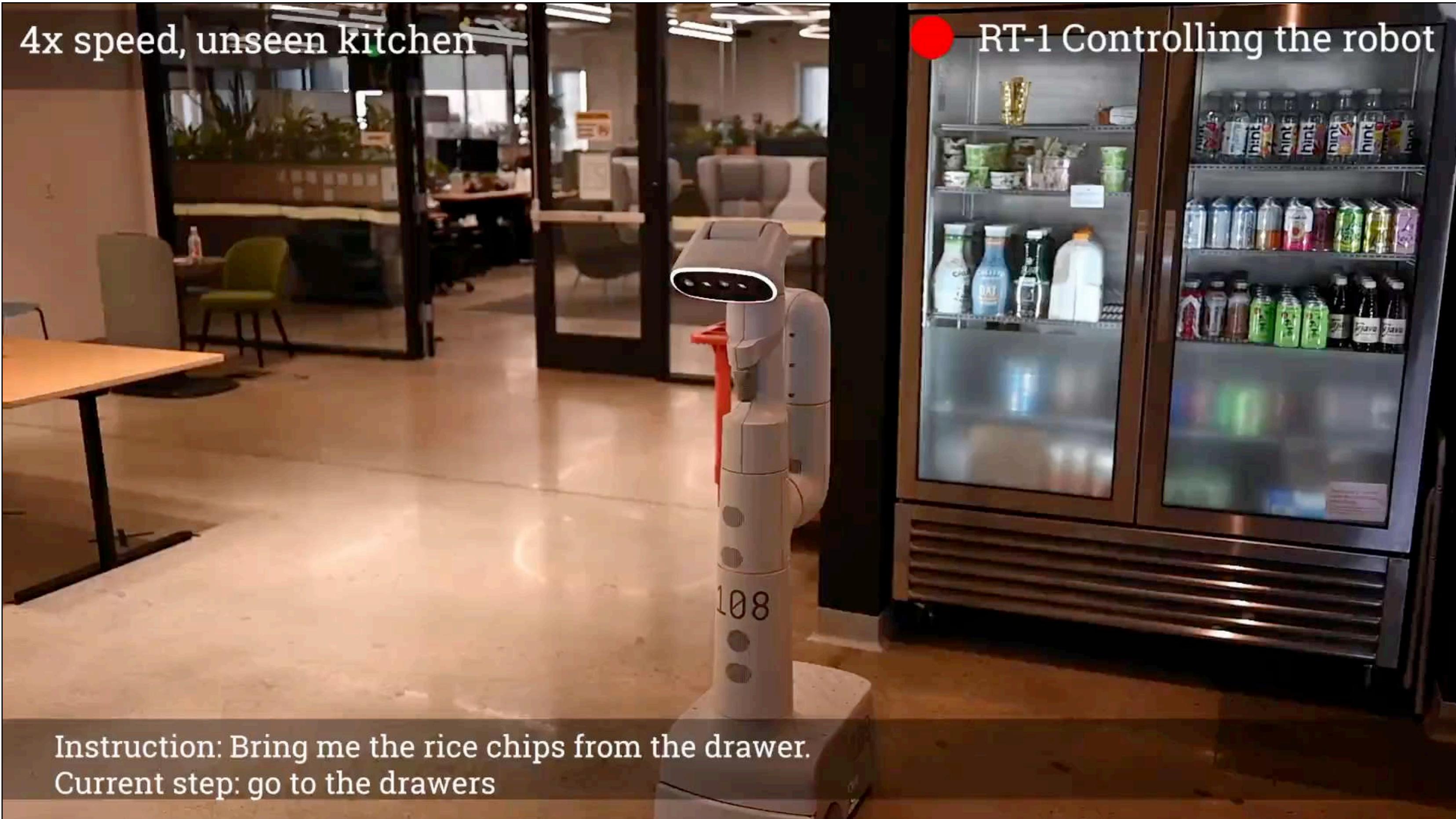


VILA running on MacBook M1 CPU

VILA: On Pre-training for Visual Language Models [Lin et al., CVPR 2024]

Vision-Language-Action Models

Robotics transformers control robots based on language instructions



Run at only **3Hz** due to the high computational cost and networking latency.

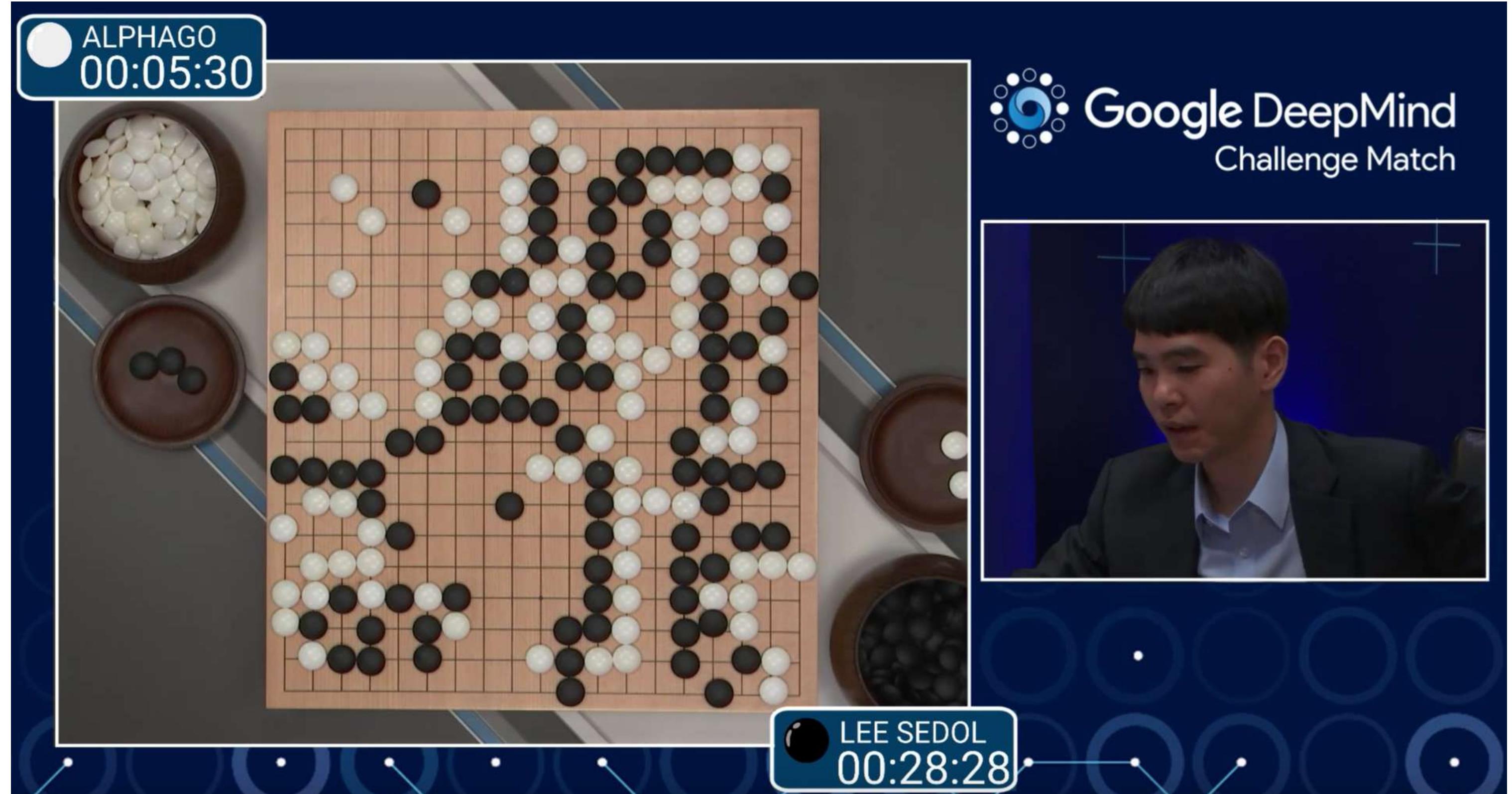
RT-1: <https://robotics-transformer1.github.io/>

Deep Learning for Games

AlphaGo masters the game of Go with DNNs & tree search



AlphaGo (Nature 2016)



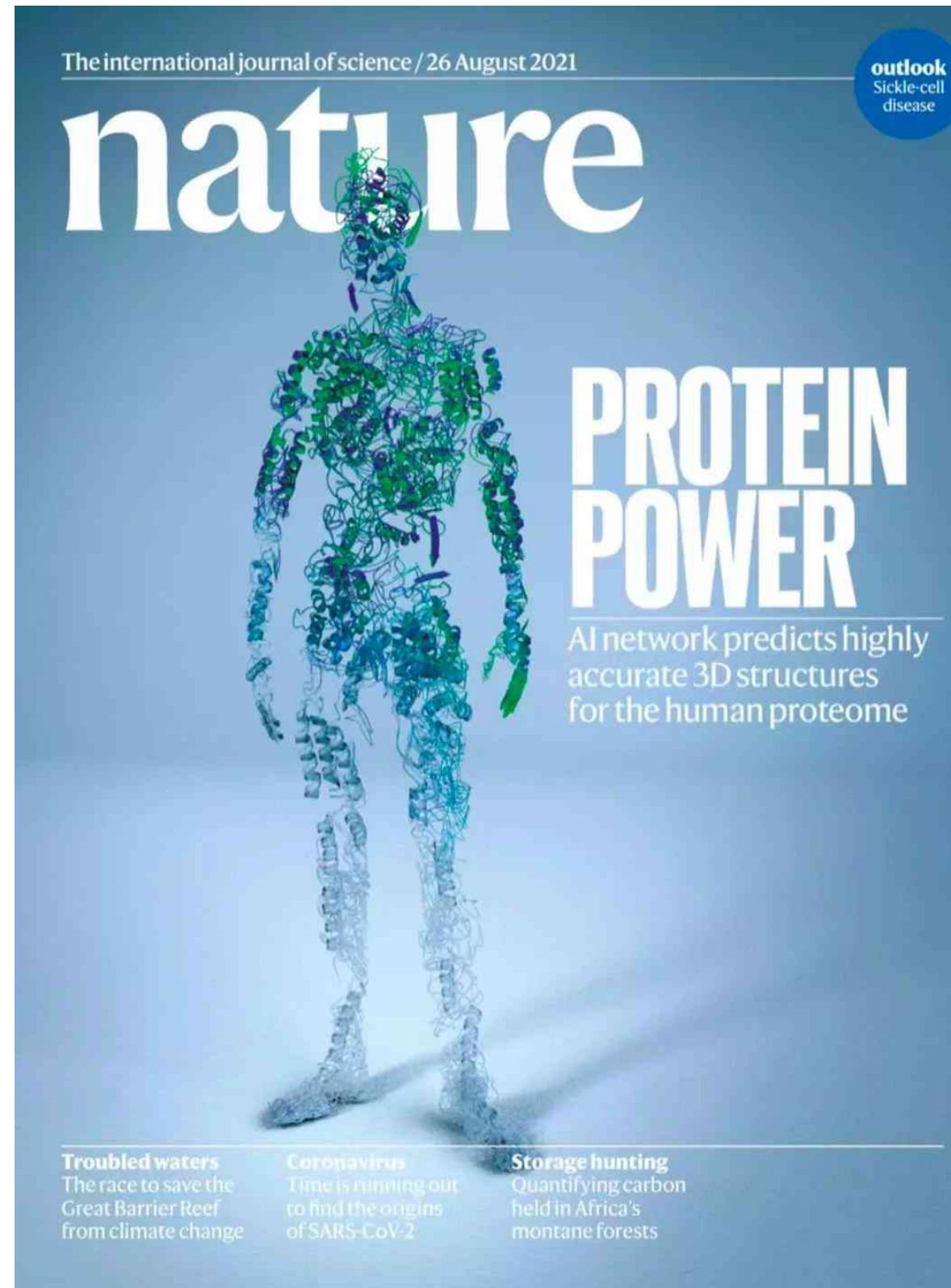
AlphaGo versus Lee Sedol (4-1)

Compute: 1920 CPUs and 280 GPUs (**\$3000 electric bill per game**)

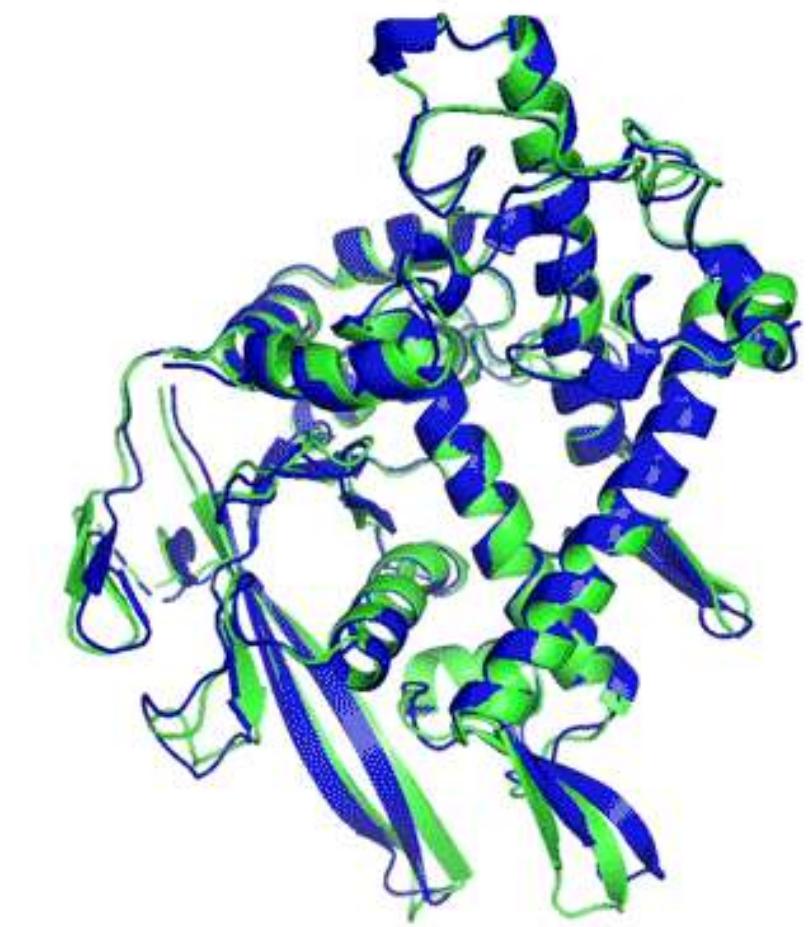
AlphaGo: <https://www.deepmind.com/research/highlighted-research/alphago>

Deep Learning for Scientific Discovery

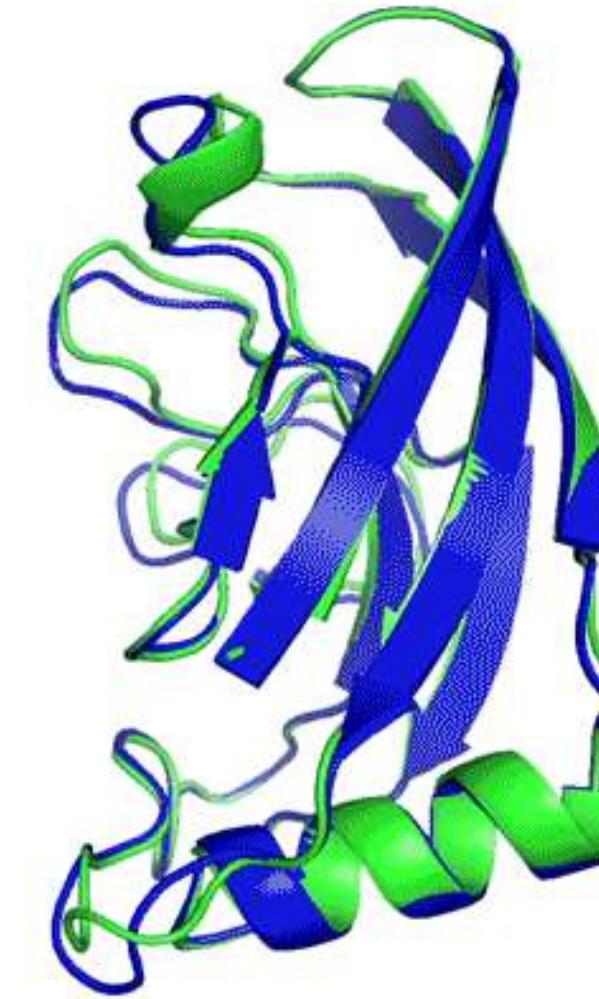
AlphaFold reveals the structure of the protein universe



AlphaFold (Nature 2021)



T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)



T1049 / 6y4f
93.3 GDT
(adhesin tip)



Experimental result



Computational prediction

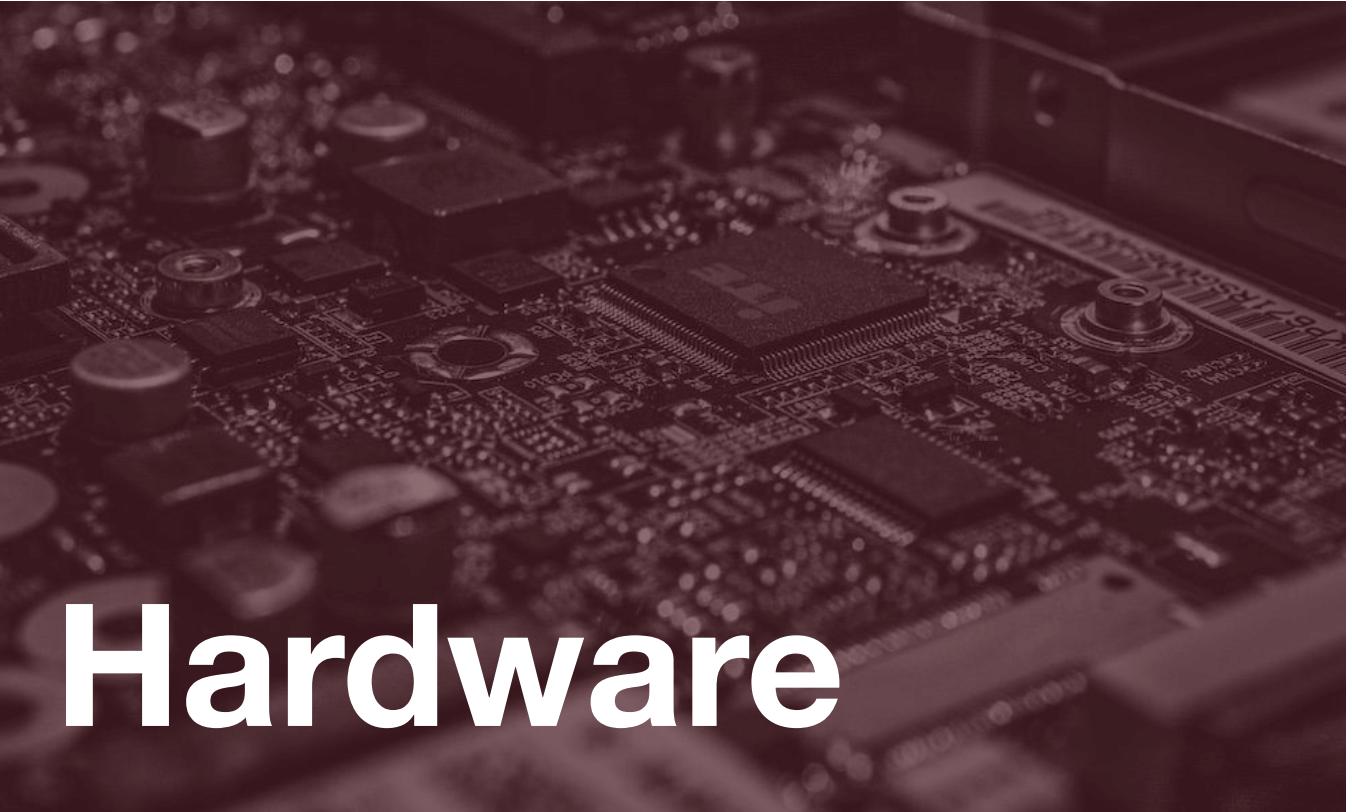
Compute: 16 TPUv3s (128 TPUv3 cores) for a few weeks

AlphaFold 2: <https://www.deepmind.com/blog/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology>

Deep Learning: Three Pillars

```
19    temp = s1[i];
20    unsigned int len1 = s1.size();
21    const size_t len2 = col(len2+1), prevCol(len2+1);
22    vector<unsigned int> col(len2+1), prevCol(len2+1);
23    for (unsigned int i = 0; i < prevCol.size(); i++) {
24        for (unsigned int j = i; j < len1; j++) {
25            col[j] = i+1;
26            for (unsigned int k = j+1; k < len2; k++) {
27                col[k] = std::min(prevCol[j] + (s1[i]==s2[k]), col[k]);
28                if (col[k] > len2) swap(prevCol);
29            }
30        }
31    }
32    return col;
33 }
```

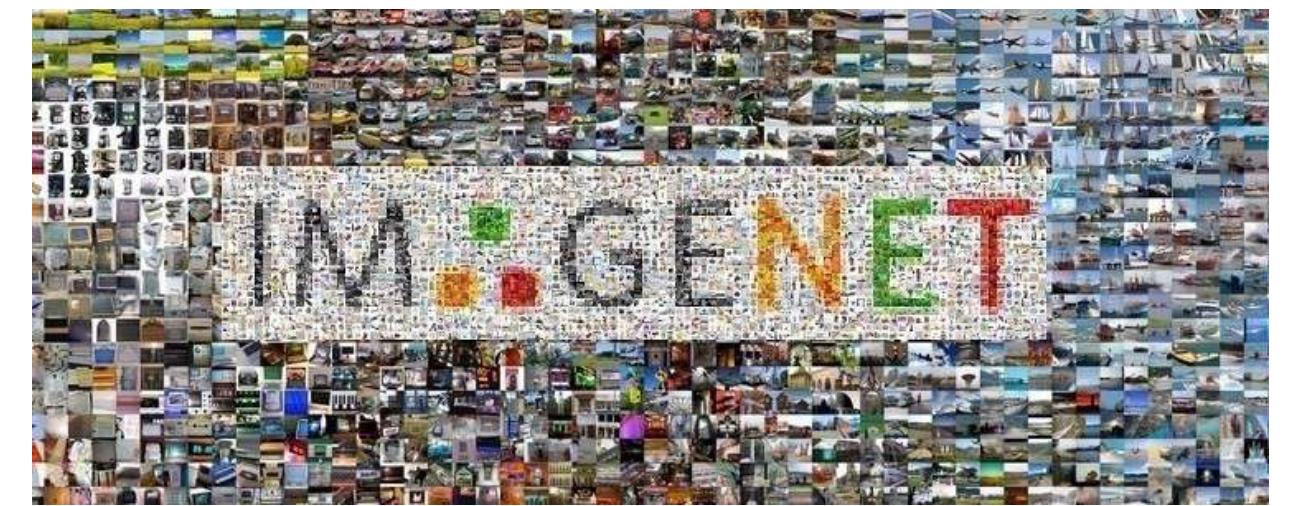
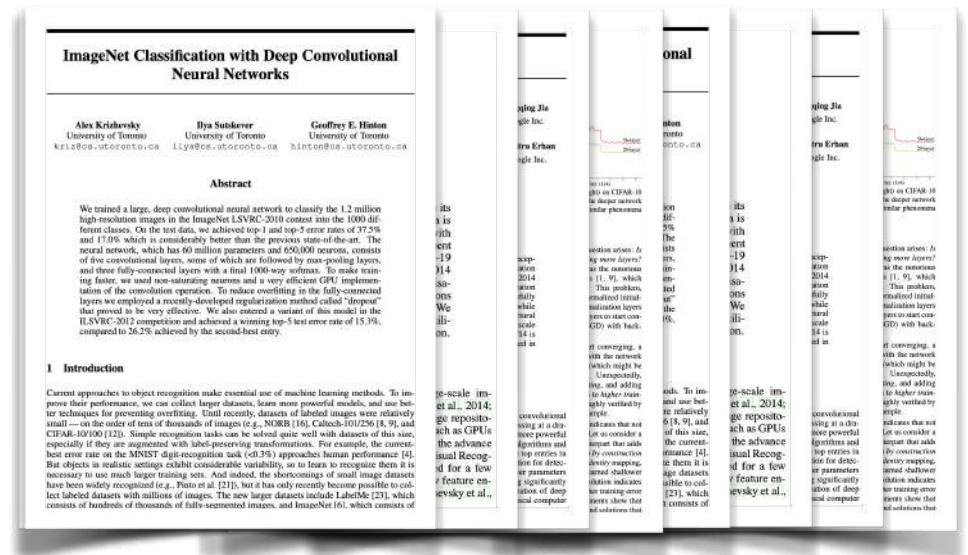
Algorithm



Hardware

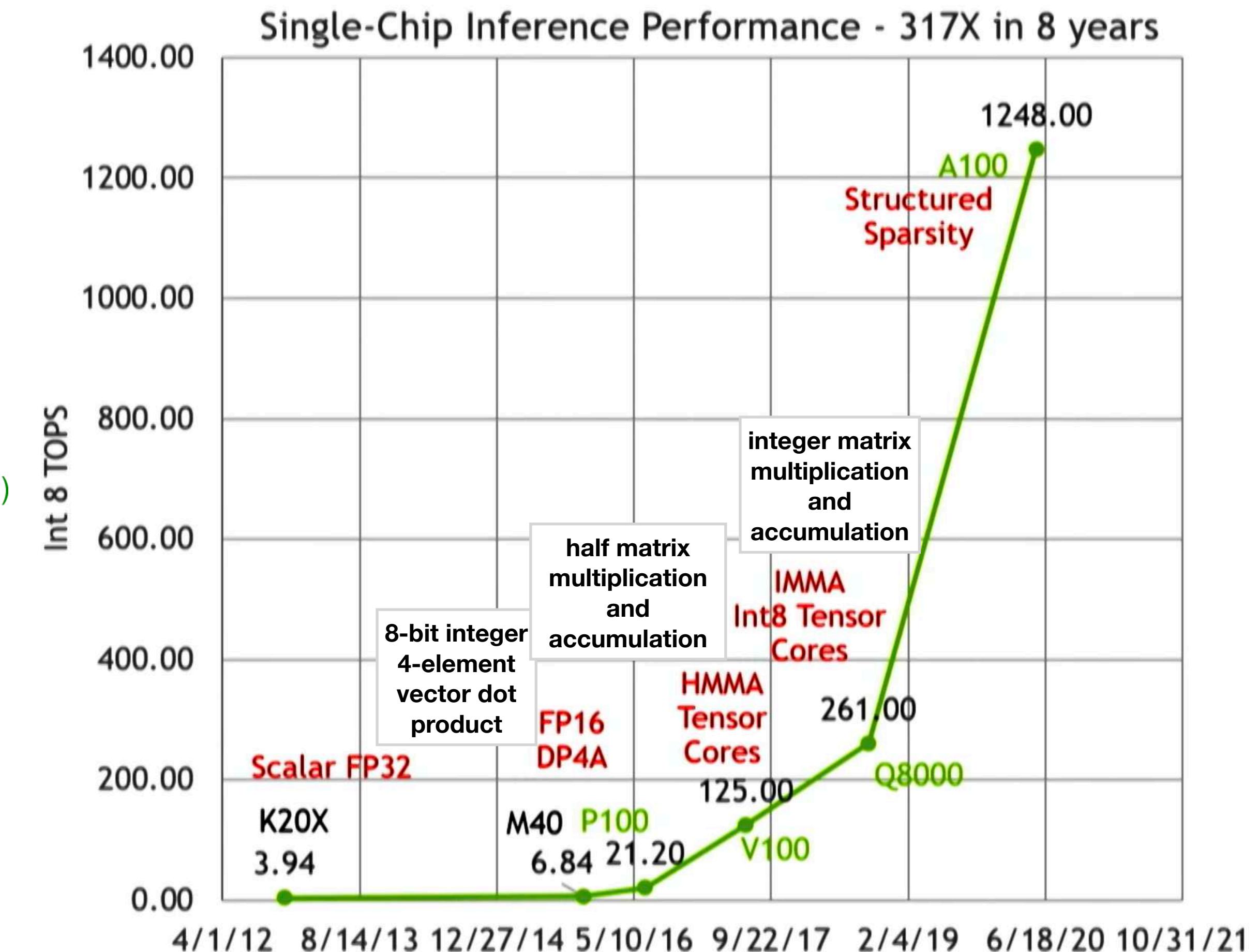
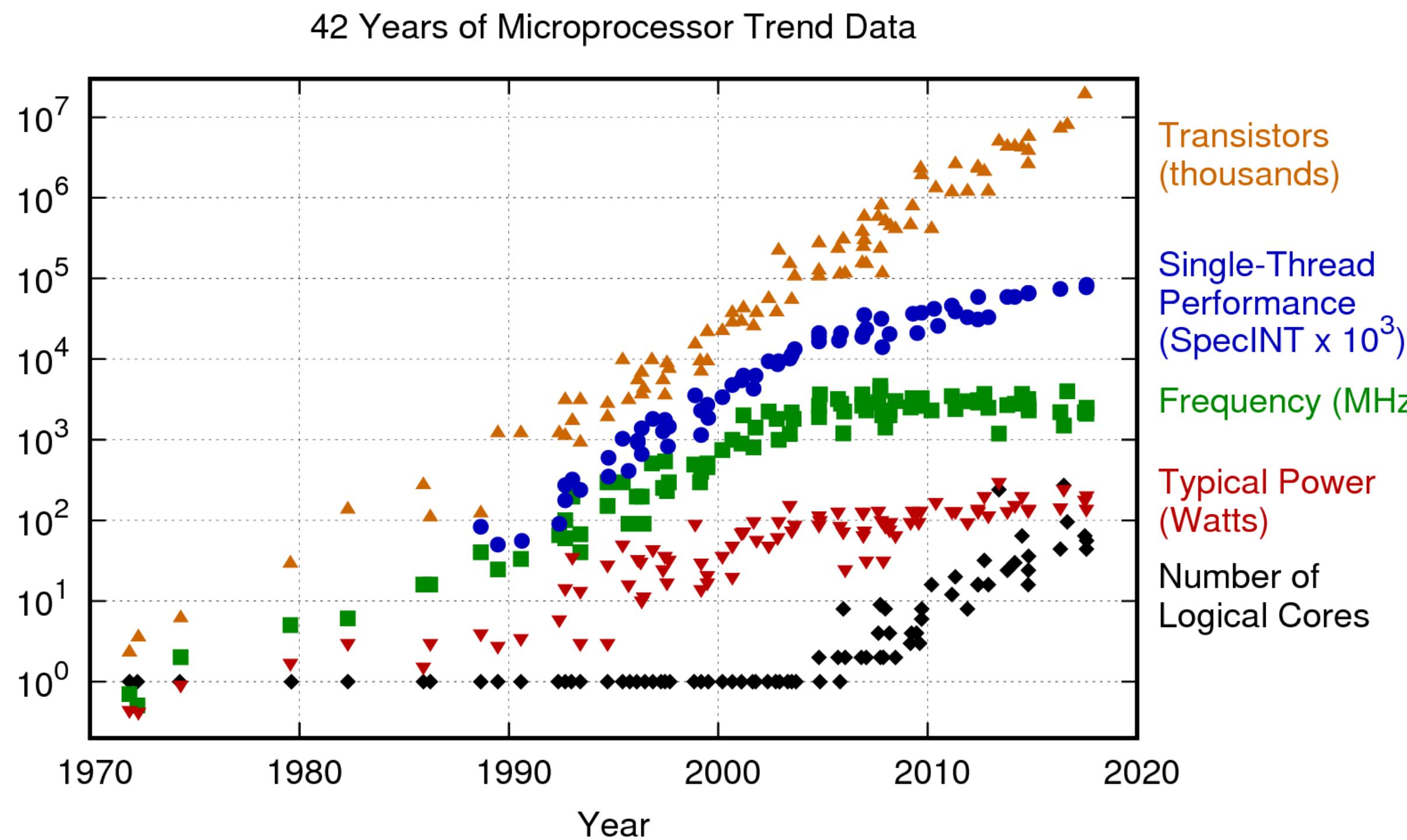


Data



Architectural support for quantization/pruning brings tremendous improvement

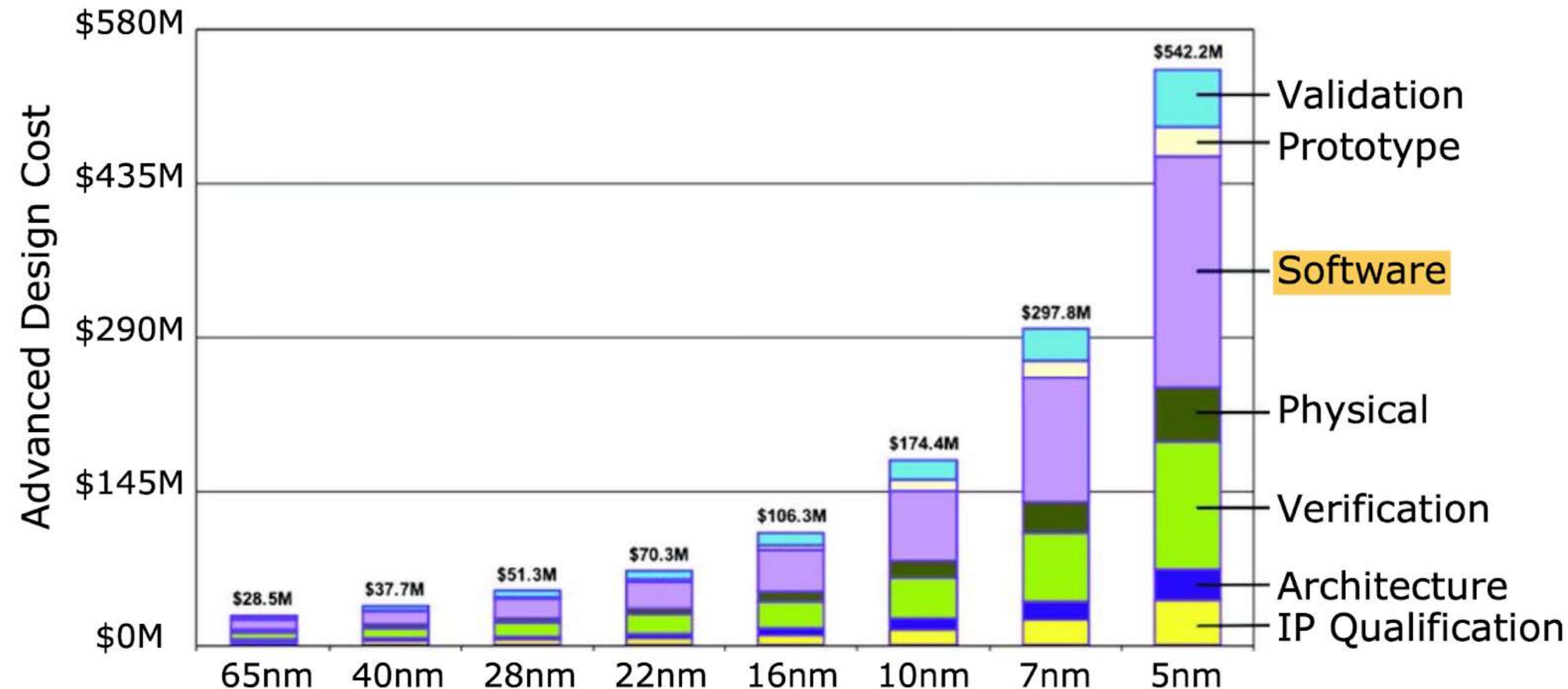
FP32 => FP16 => Int8; dense => sparse



Data source: M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, C. Batten, K. Rupp

GPUs, Machine Learning, and EDA — Bill Dally

Software Innovation is important in advanced technology node



The software cost dominates the cost breakdown of advanced technology nodes [source].
We focus on designing new algorithms and software for efficient computing.

Cloud AI Hardware

NVIDIA



P100 (2016)



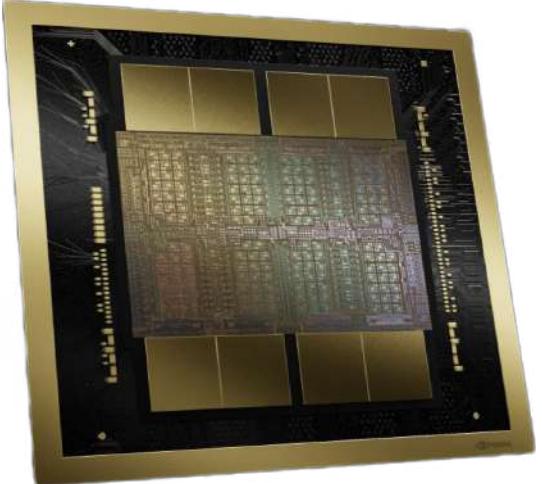
V100 (2017)



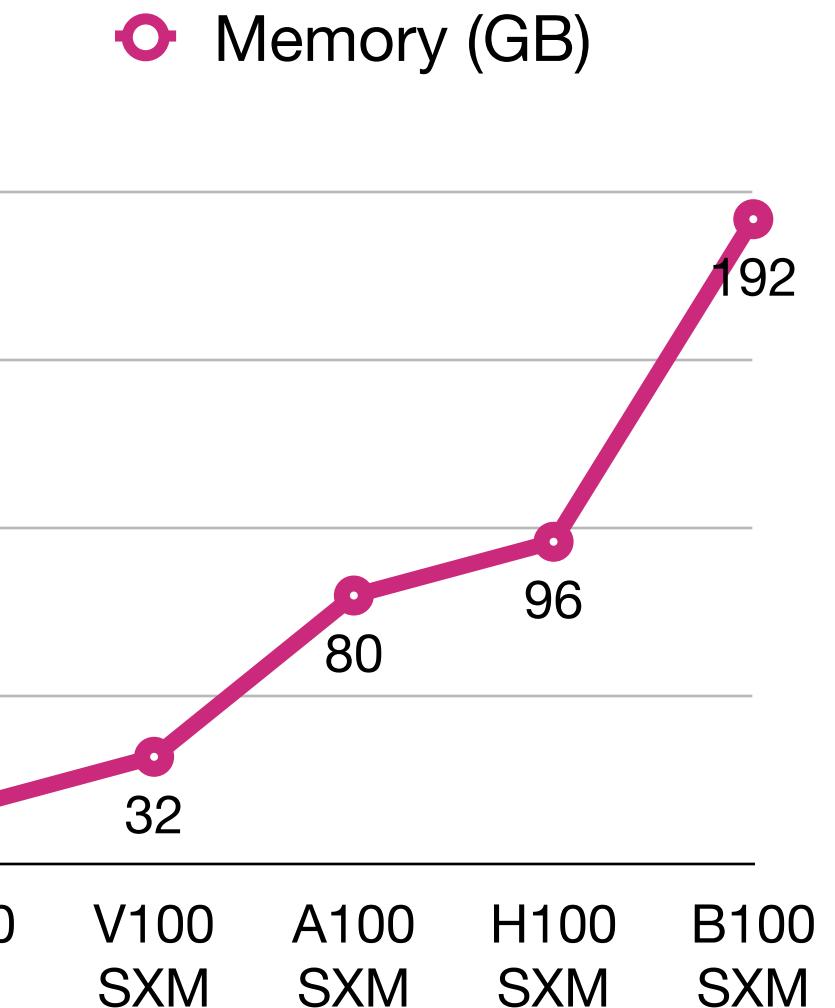
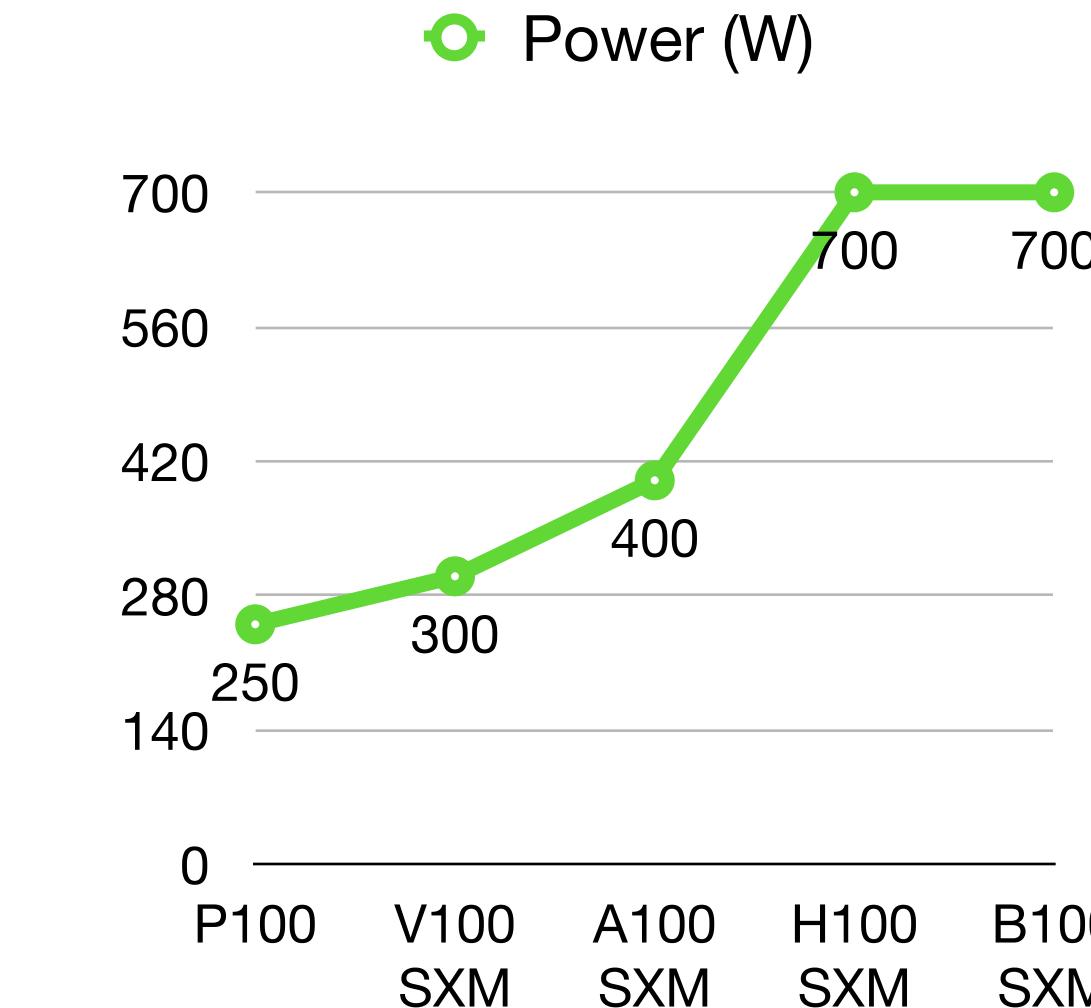
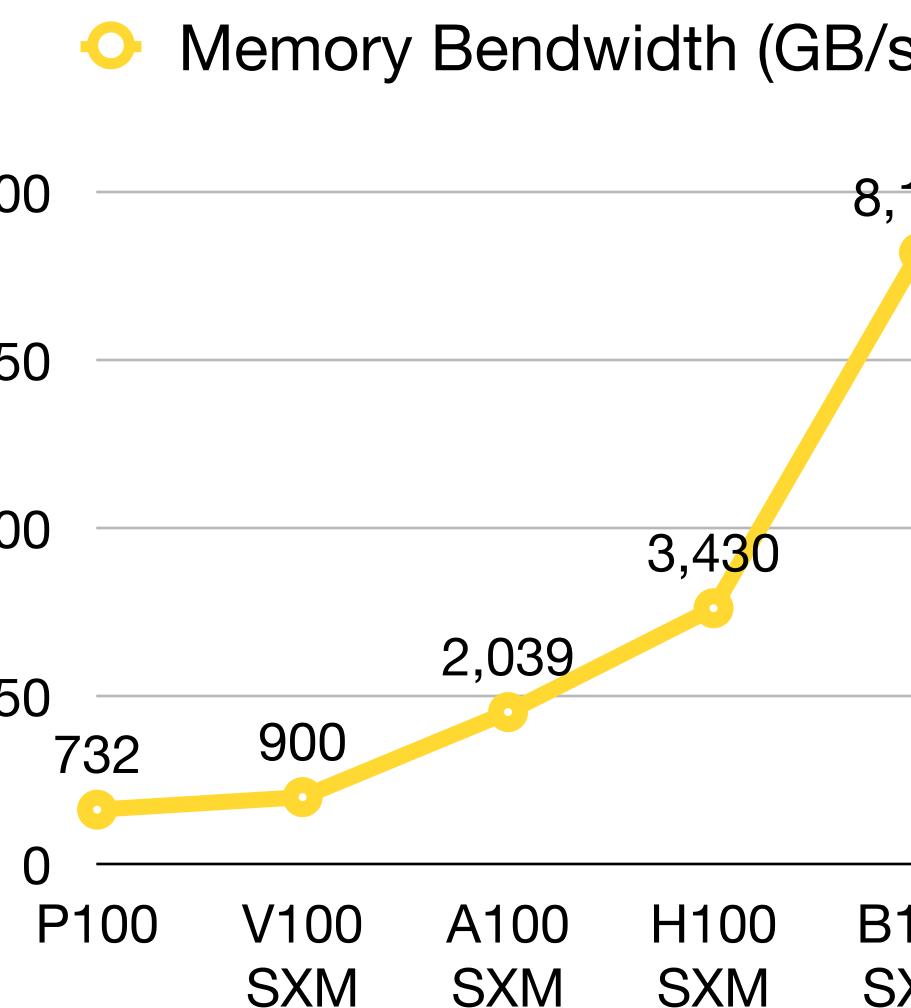
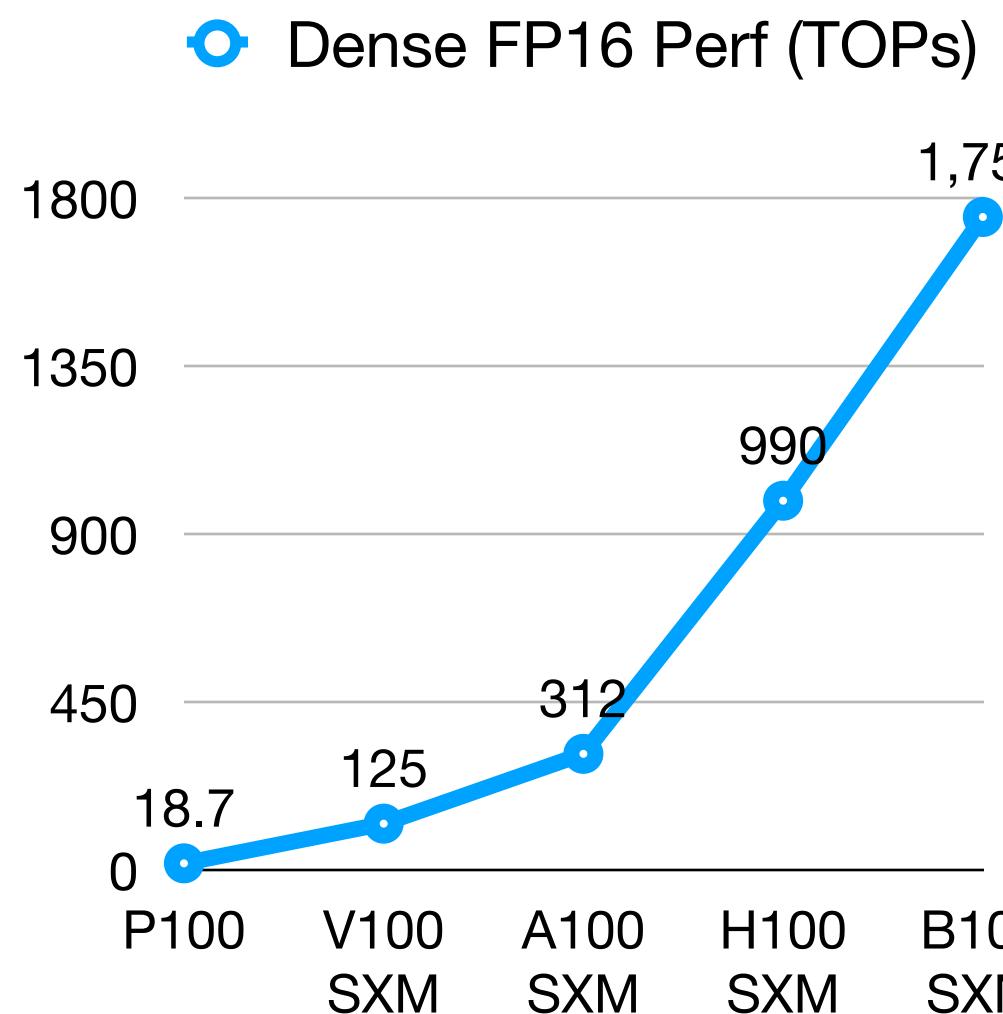
A100 (2020)



H100 (2022)



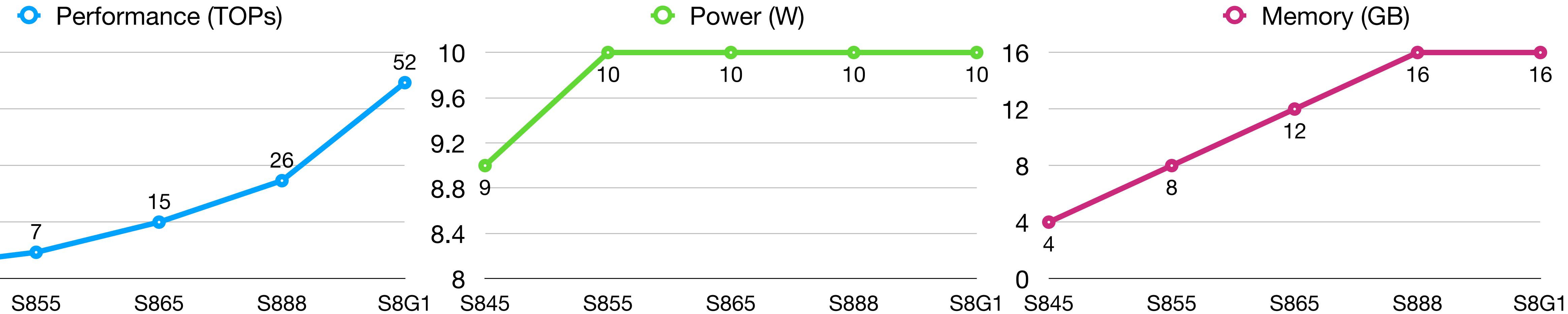
B100 (2024)



Edge AI Hardware

Qualcomm Hexagon DSP

- Qualcomm Hexagon is a family of digital signal processor (DSP) products by Qualcomm. It is designed to deliver performance with low power over a variety of applications

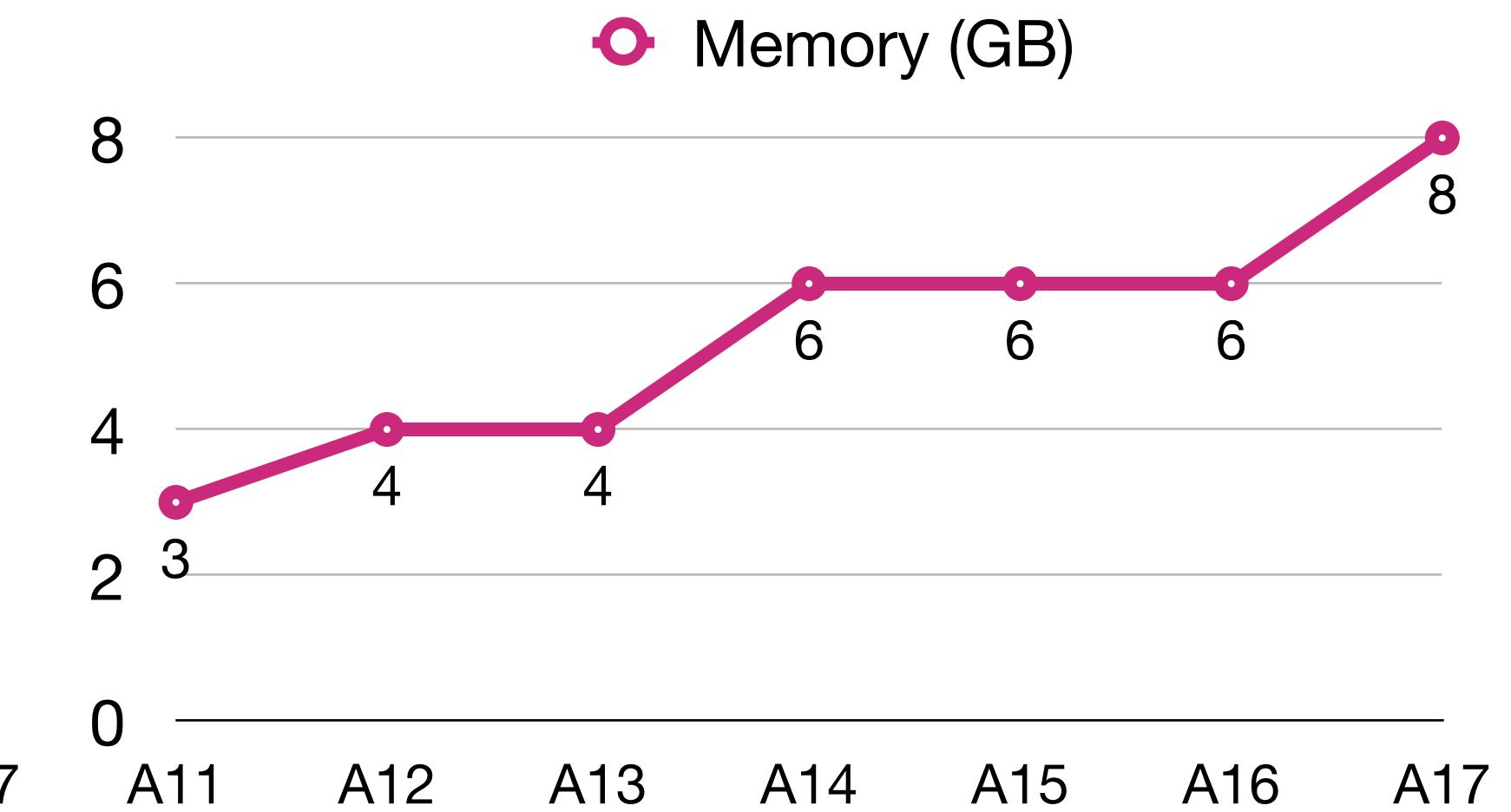
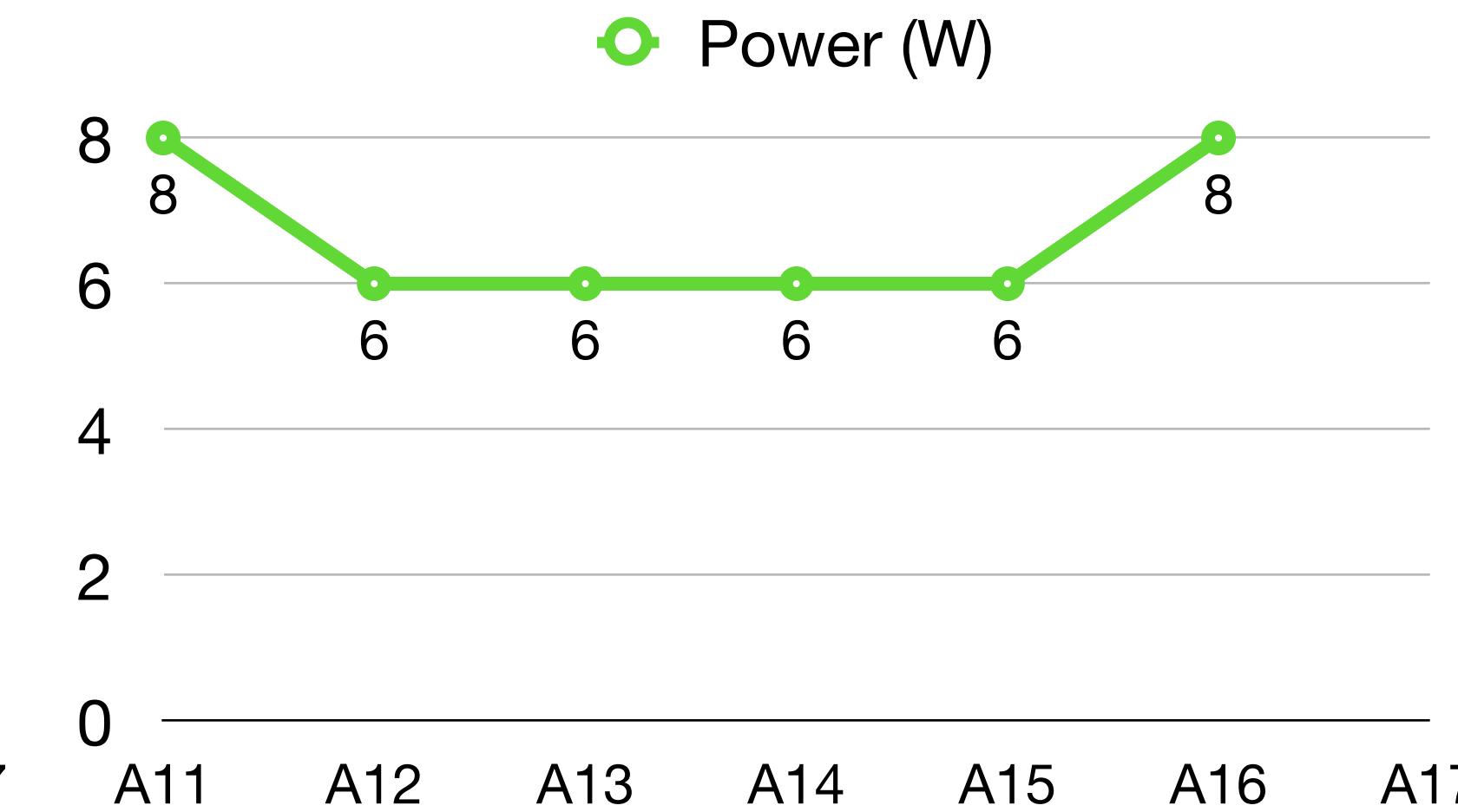
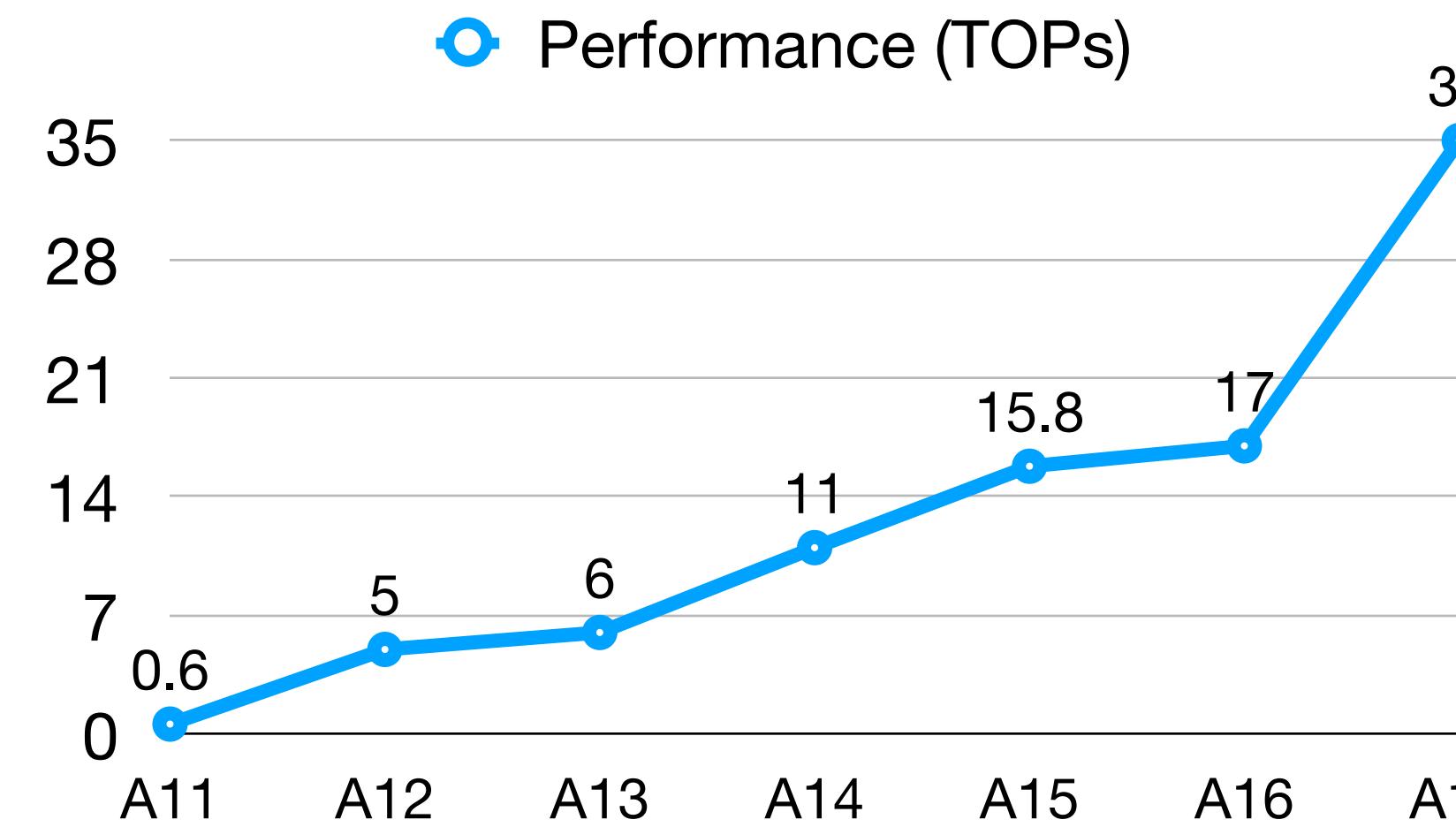


https://en.wikipedia.org/wiki/List_of_Qualcomm_Snapdragon_processors

Edge AI Hardware

Apple Neural Engine

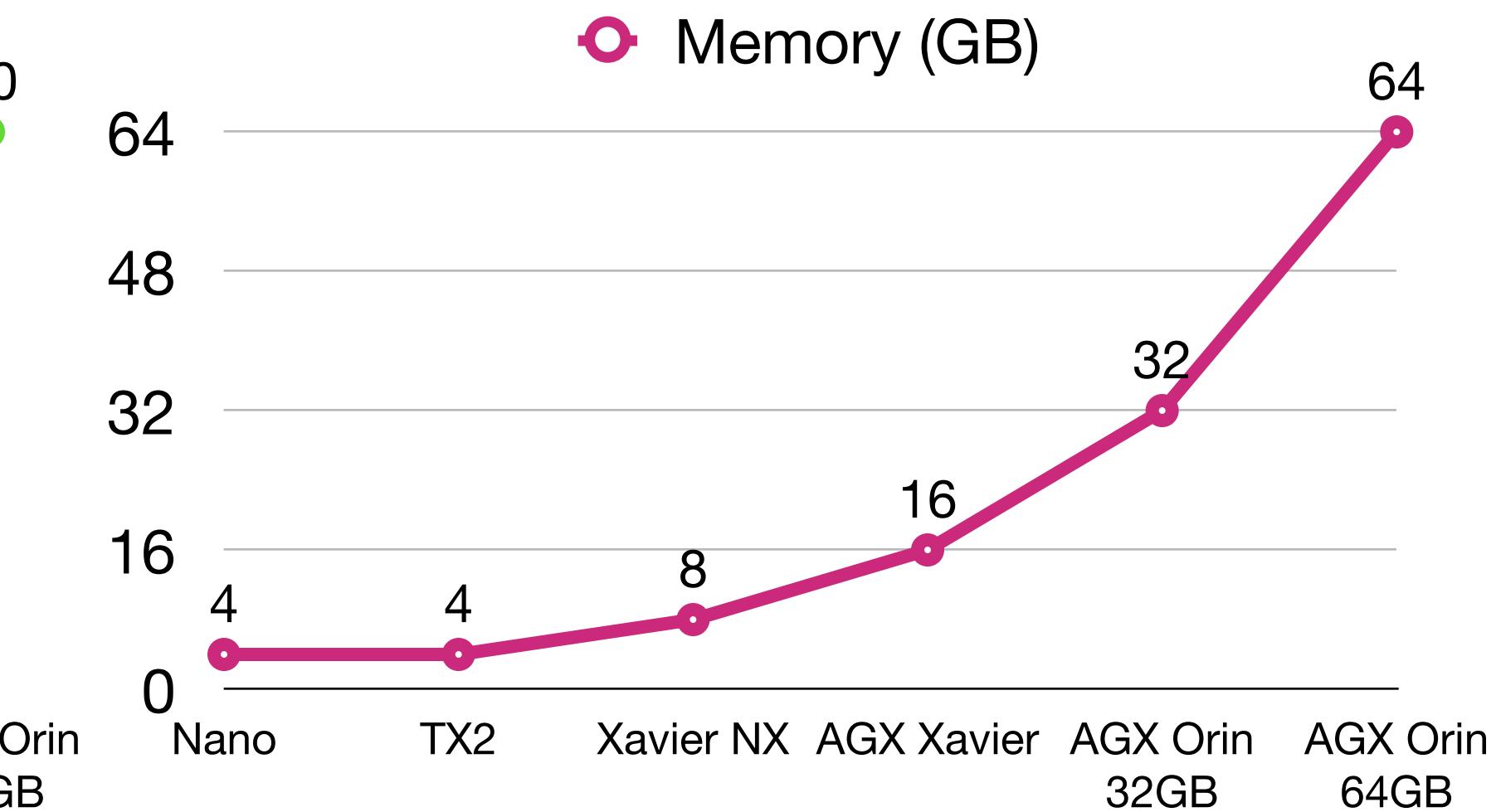
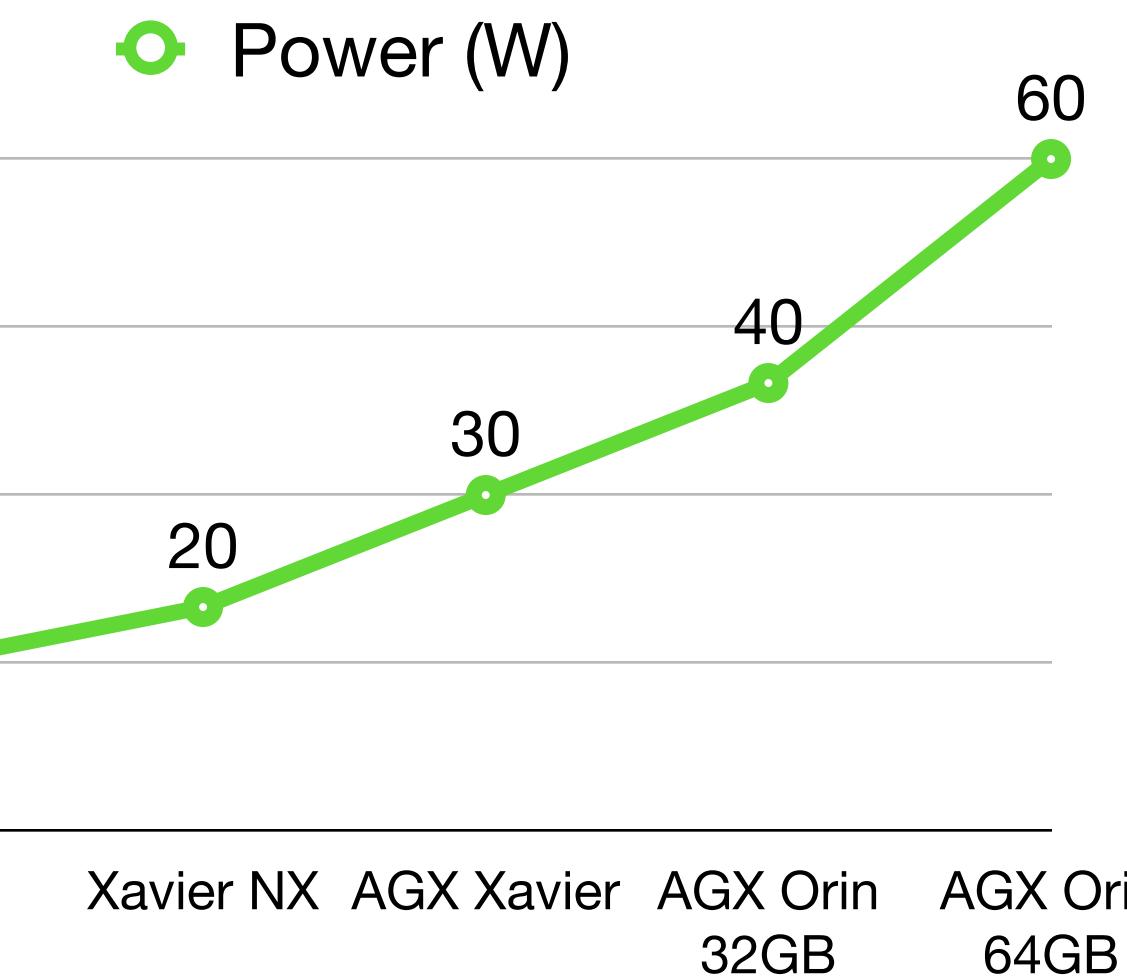
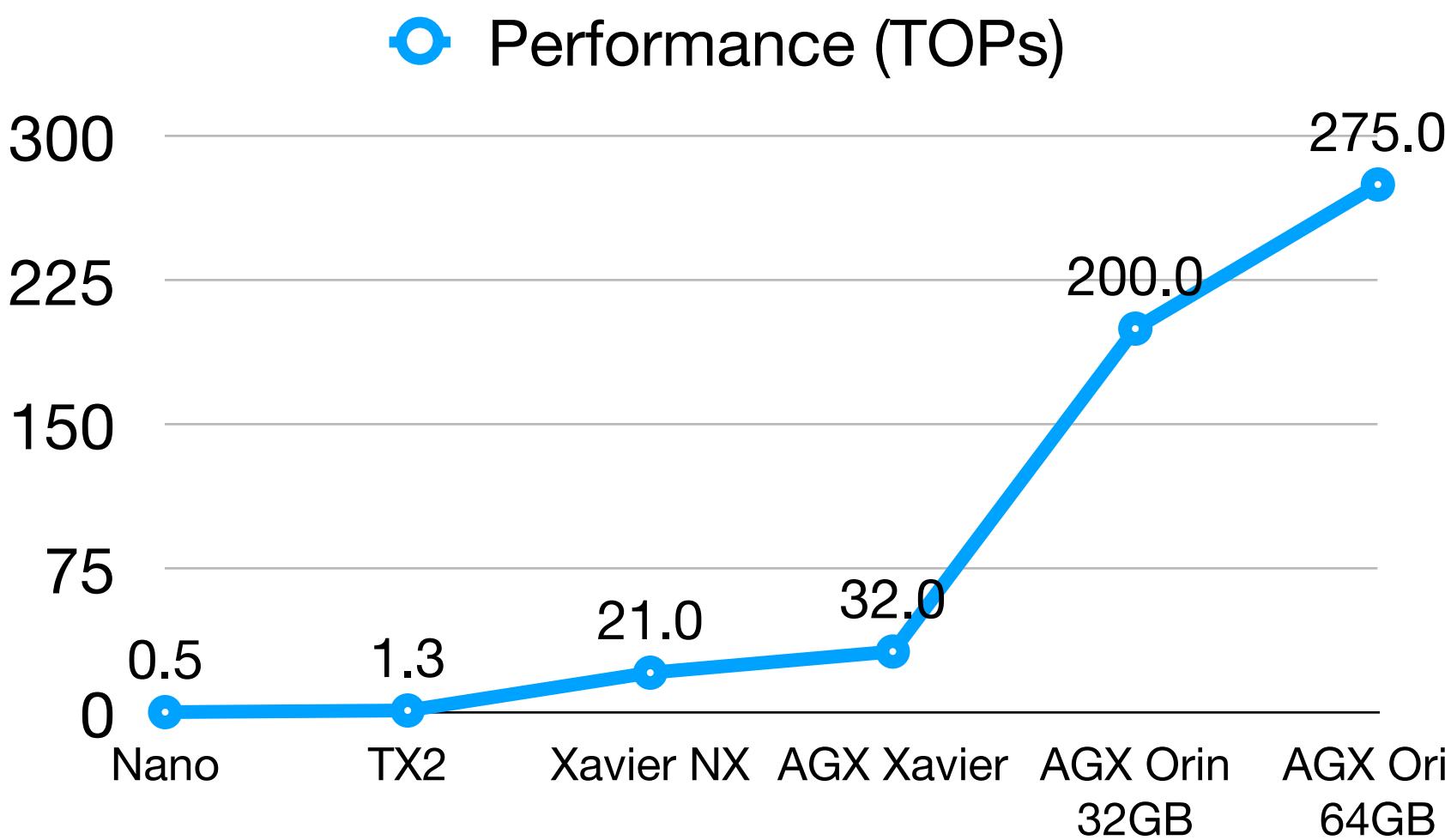
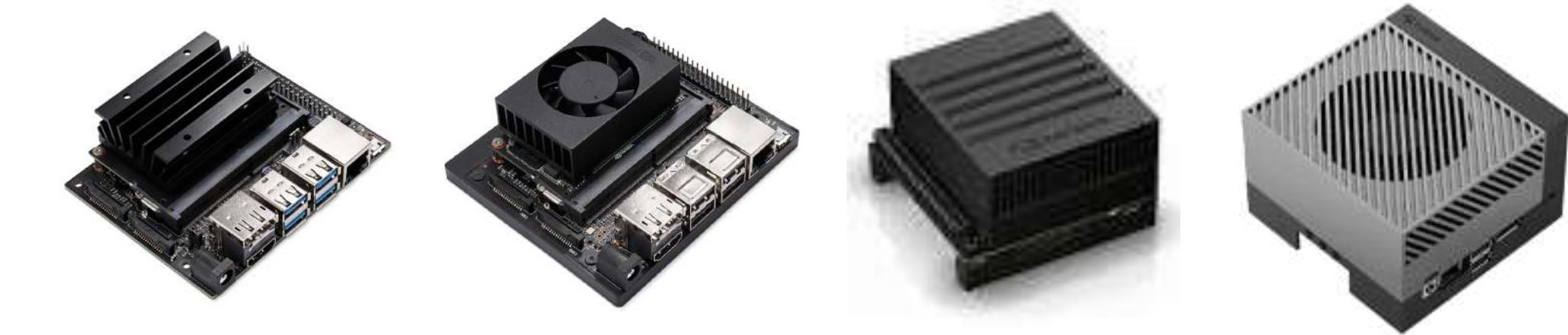
- The Apple Neural Engine (ANE) is an energy-efficient and high-throughput engine for ML inference on Apple silicon.



Edge AI Hardware

Nvidia Jetson

- NVIDIA Jetson is a complete System on Module (SOM) that includes a GPU, CPU, memory, power management, high-speed interfaces, and more.

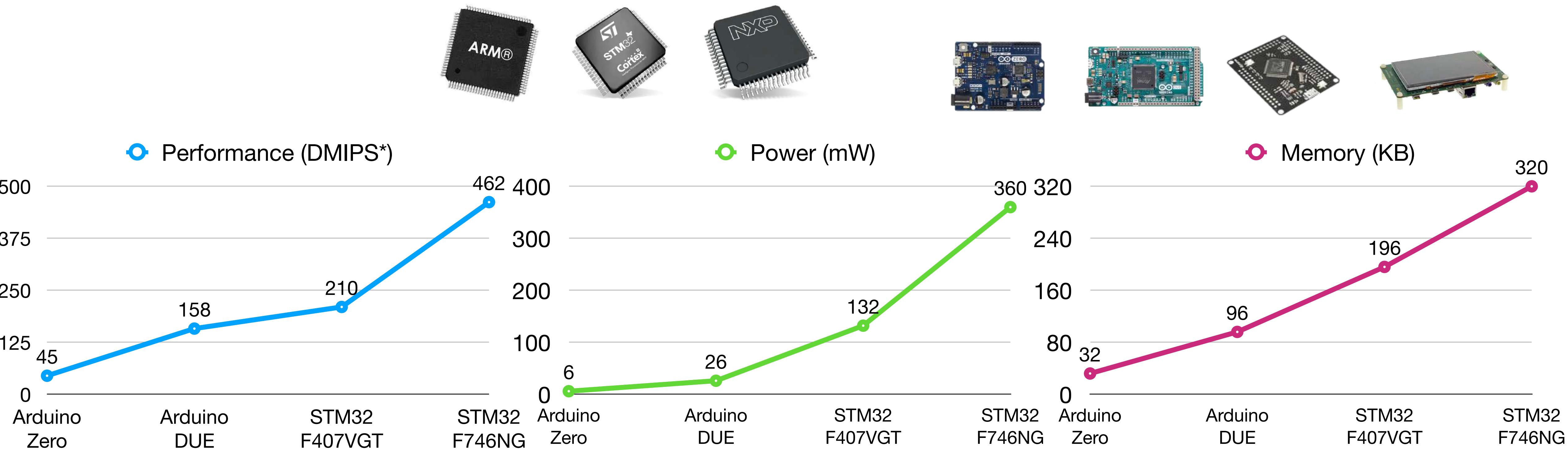


NanoReview. <https://connecttech.com/jetson/jetson-module-comparison/>

Edge AI Hardware

Microcontrollers (MCU)

- A microcontroller is a compact integrated circuit designed for embedded systems. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.



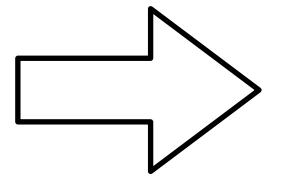
* Dhrystone Million Instructions Per Second (DMIPs) is an index for integer computation.

Edge AI Hardware

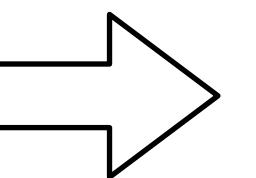
Edge AI devices still have huge gap to cloud processors



Cloud AI



Mobile AI



Tiny AI

Memory (Activation)

80GB

4GB

320kB

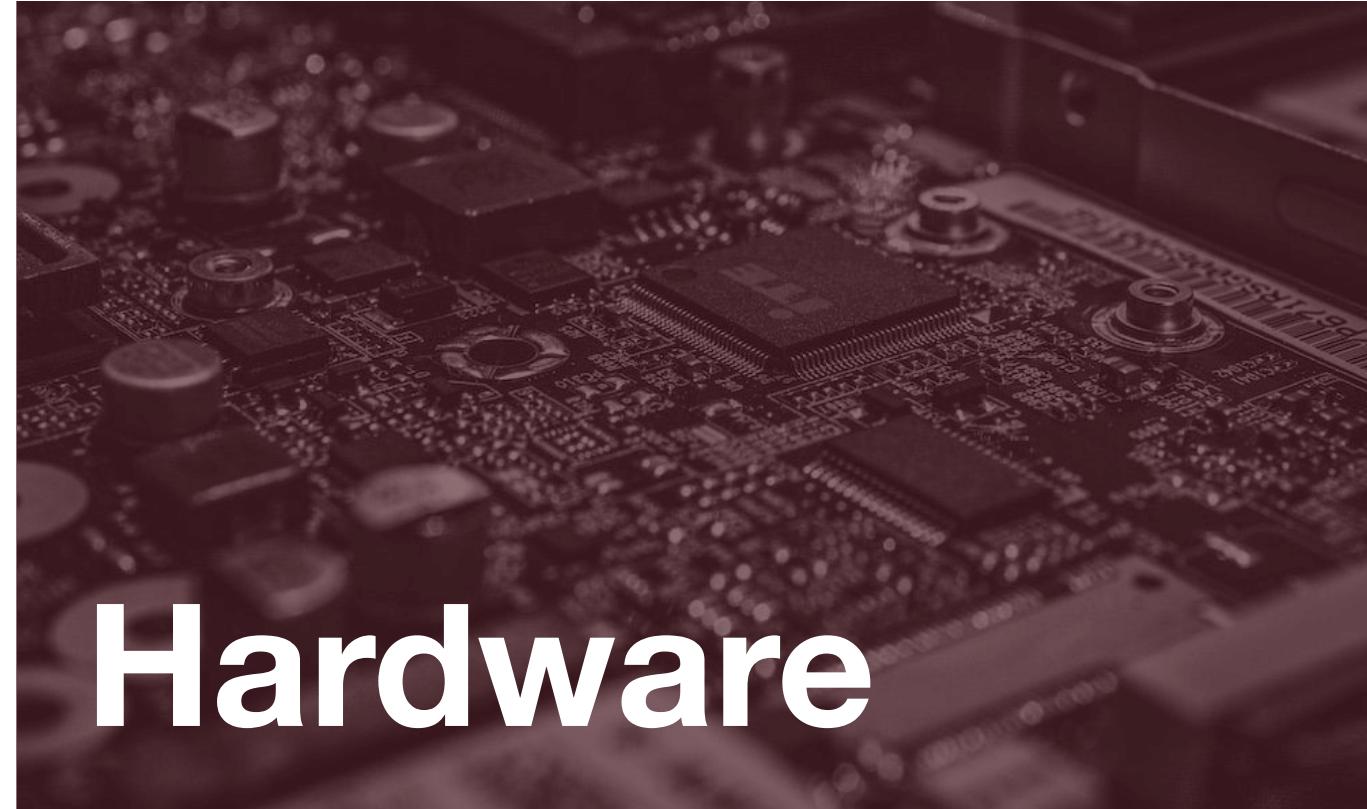
Storage (Weights)

~TB/PB

256GB

1MB

Deep Learning: Three Pillars



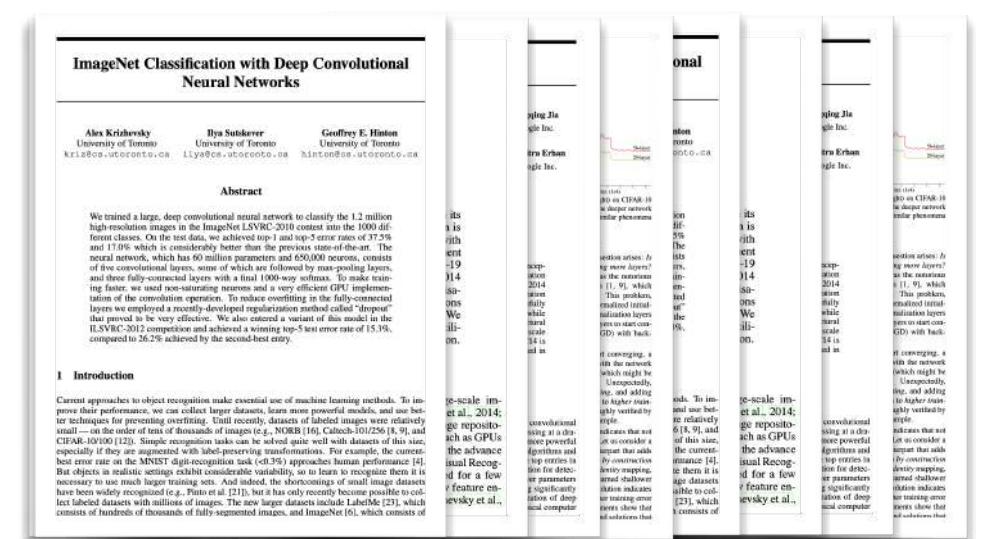
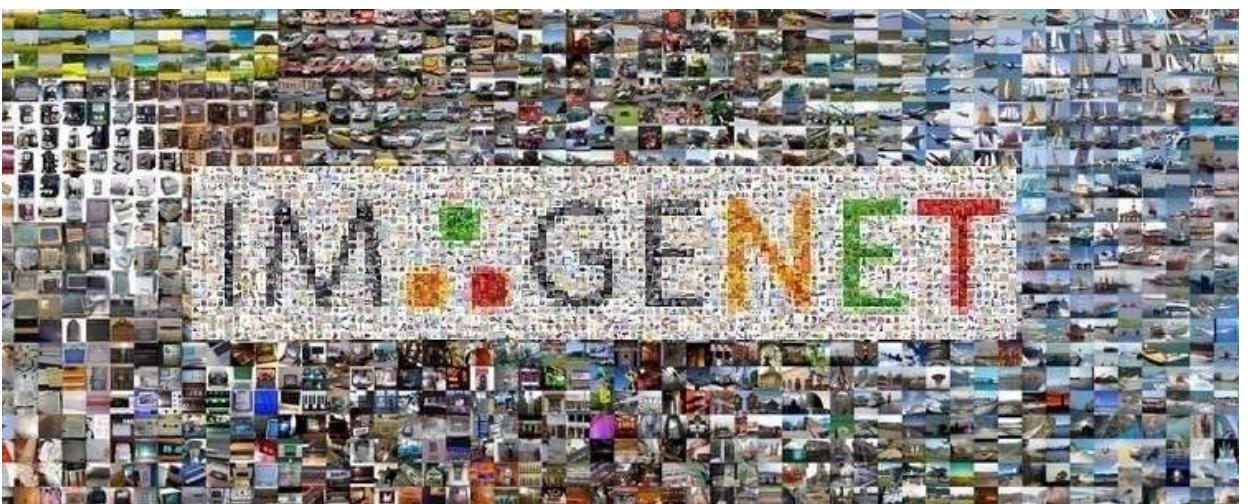
Hardware



Data

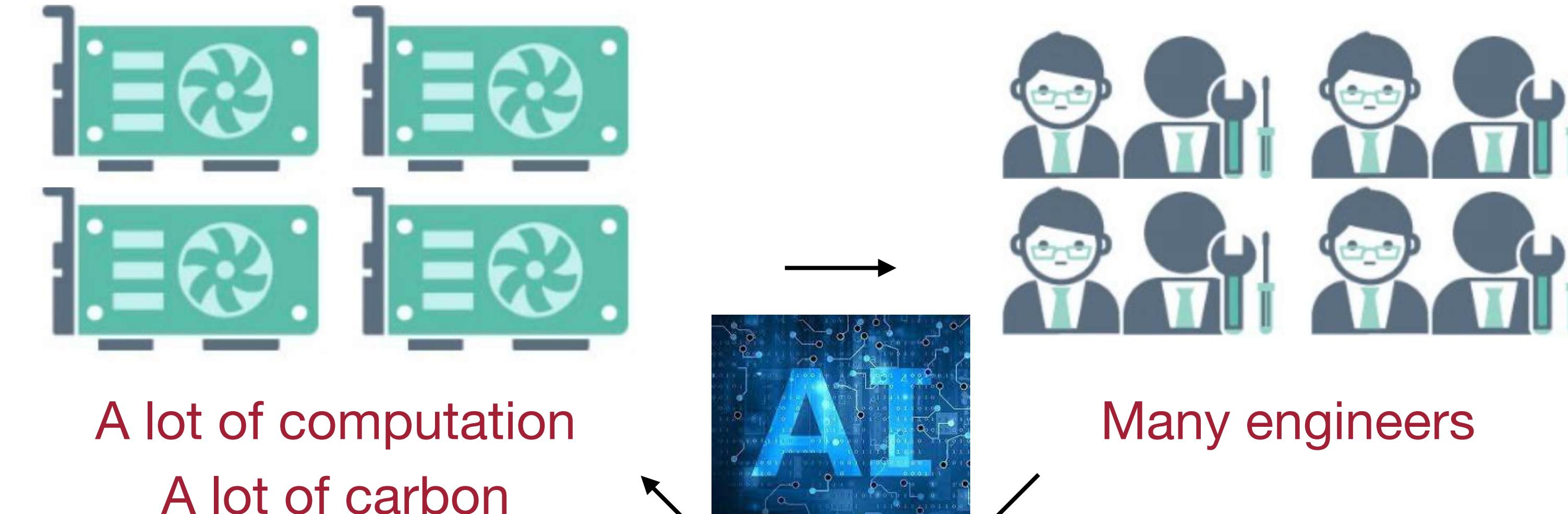


Algorithm



Current Landscape of Deep Learning

Big computation, engineer and data



Tiny Deep Learning

Tiny computation, engineer and data



less computation
less carbon

TinyML



fewer engineers



less data

Course Overview

Course Overview



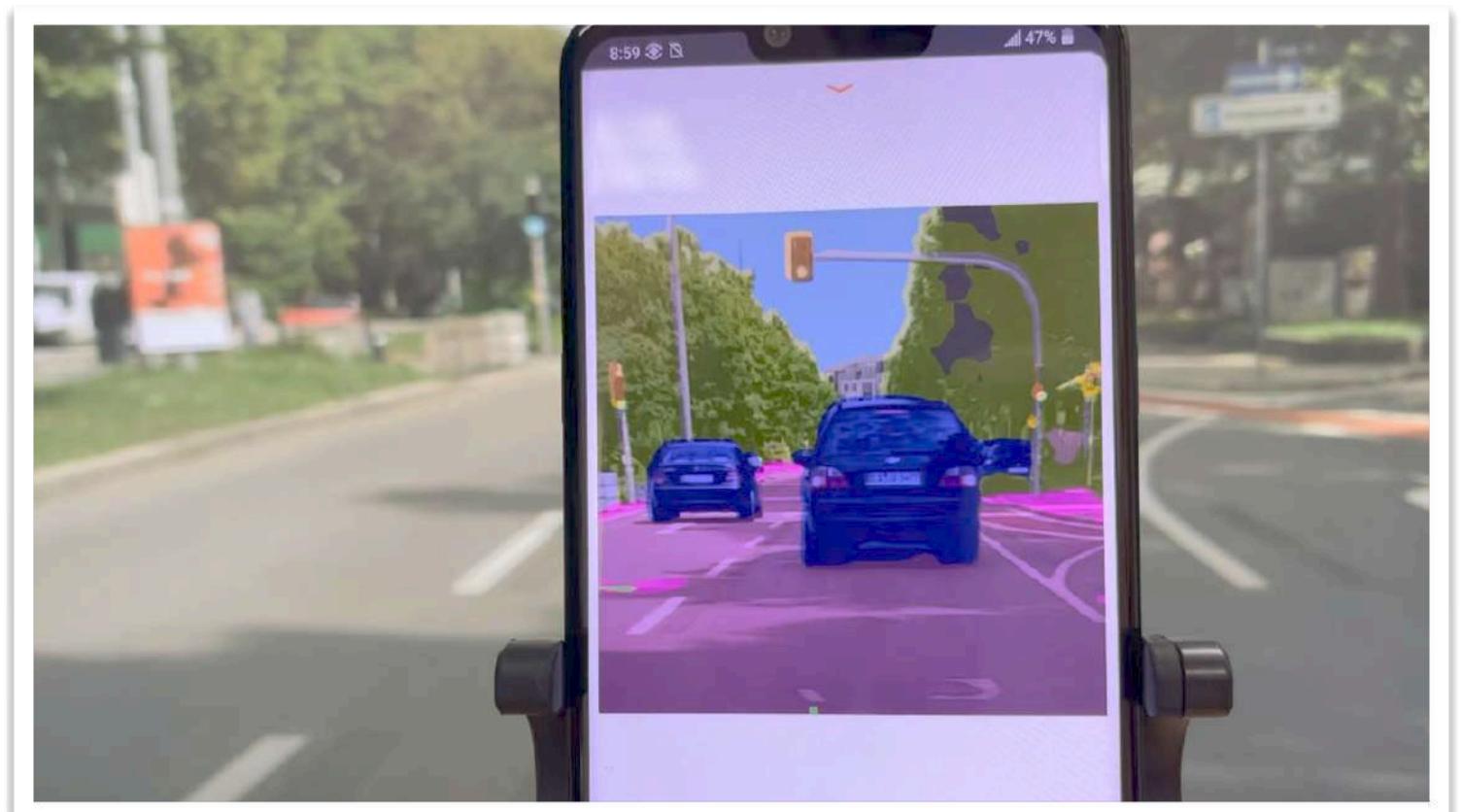
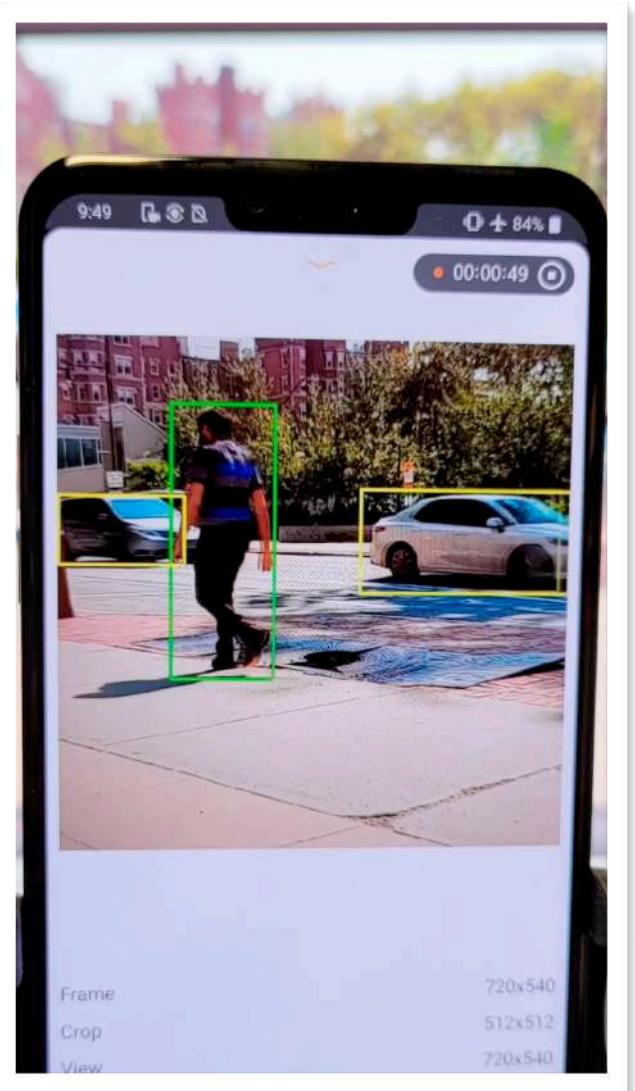
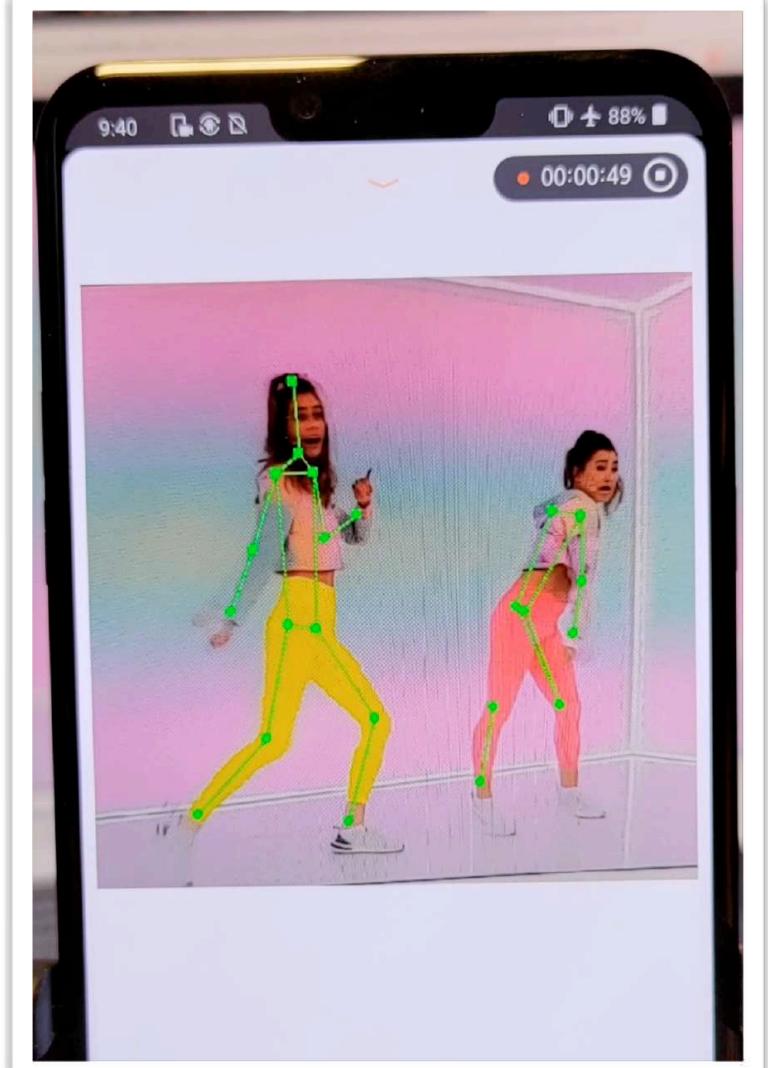
TinyML and Efficient Deep Learning Computing

6.5940 • Fall • 2024 • <https://efficientml.ai>

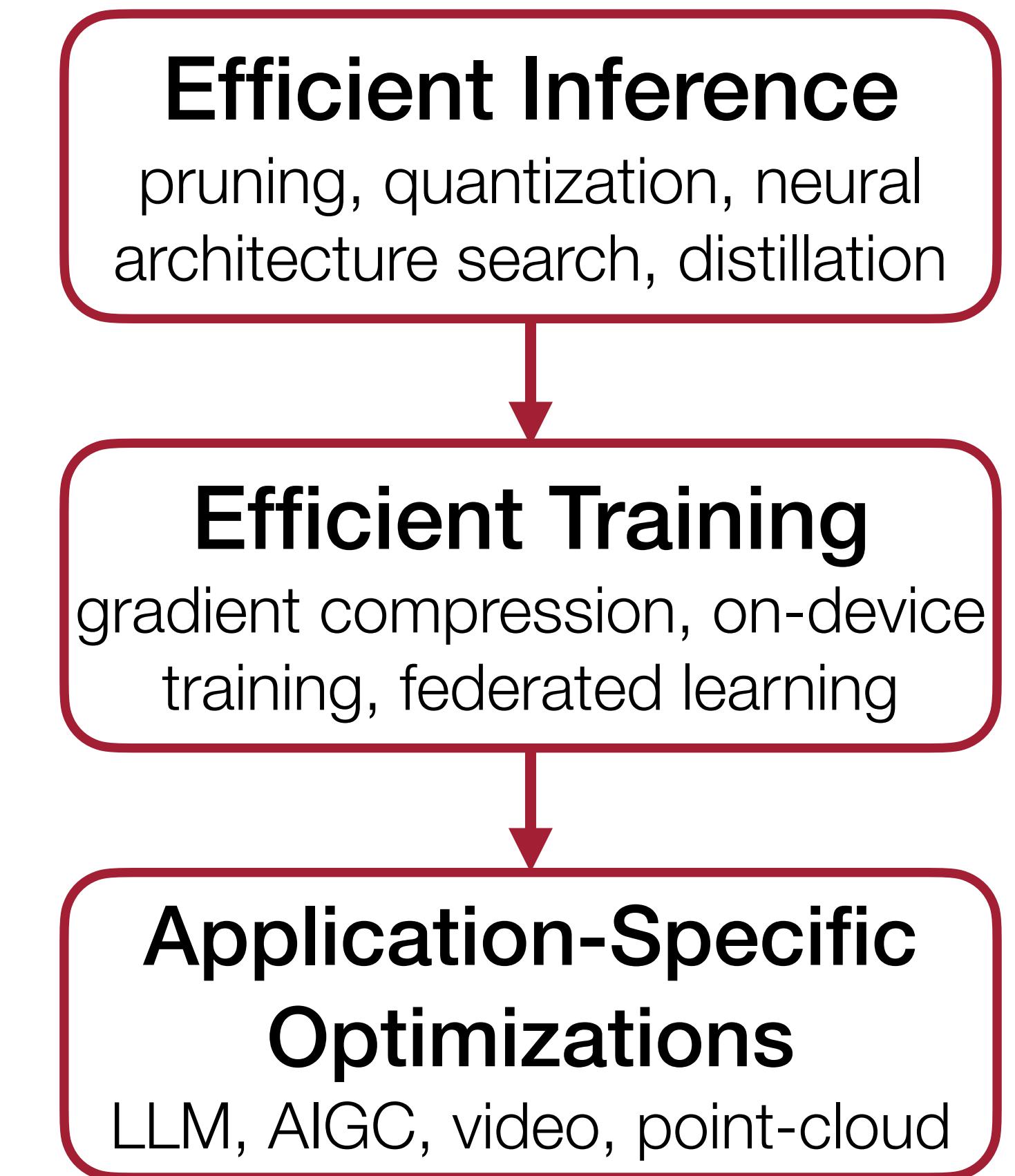
About Logistics All Courses

This course focuses on efficient machine learning and systems. This is a crucial area as deep neural networks demand extraordinary levels of computation, hindering its deployment on everyday devices and burdening the cloud infrastructure. This course introduces efficient AI computing techniques that enable powerful deep learning applications on resource-constrained devices. Topics include model compression, pruning, quantization, neural architecture search, distributed training, data/model parallelism, gradient compression, and on-device fine-tuning. It also introduces application-specific acceleration techniques for large language models and diffusion models. Students will get hands-on experience implementing model compression techniques and deploying large language models (Llama2-7B) on a laptop.

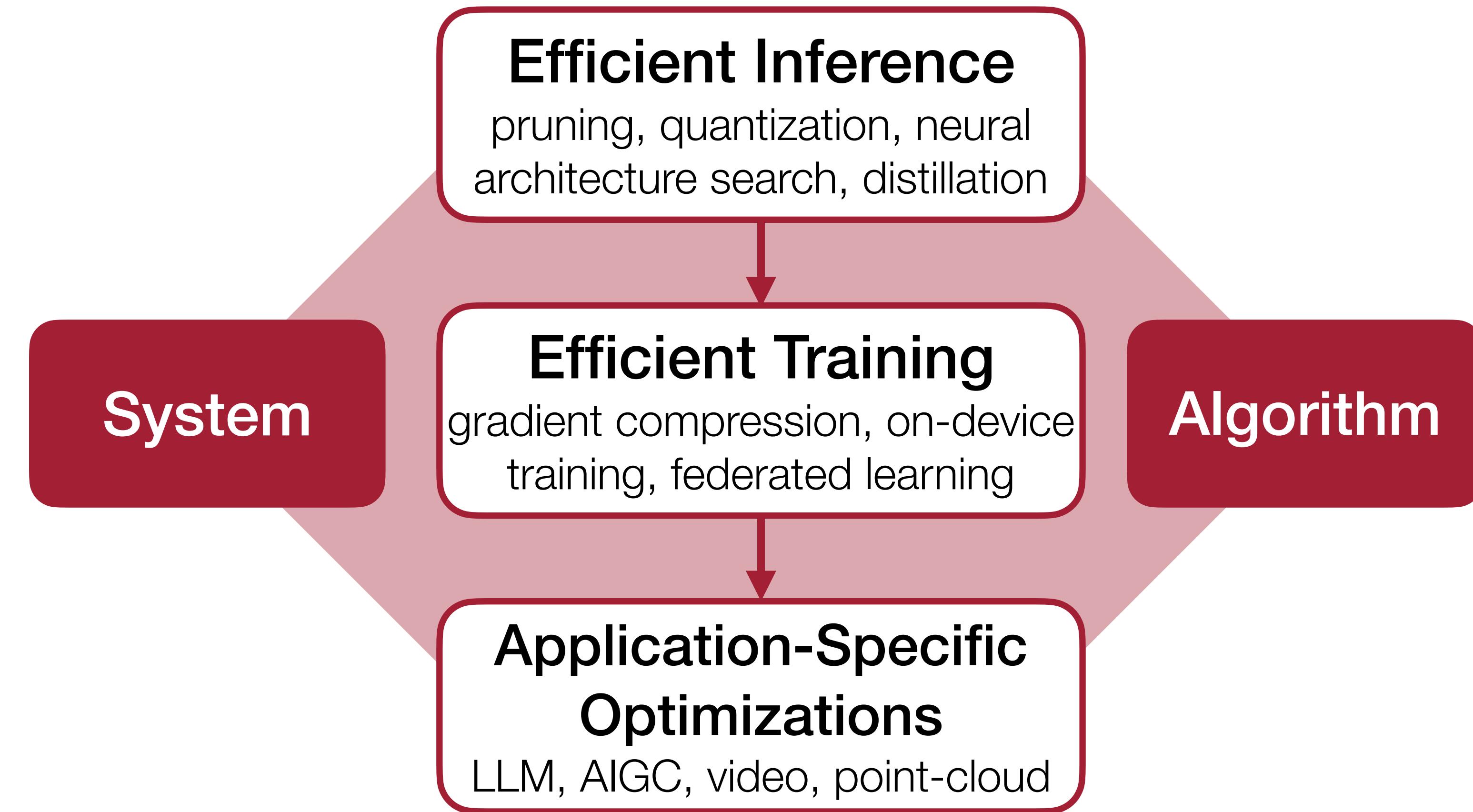
- **Lecture Videos:** <https://live.efficientml.ai/>
- **Time:** Tuesday/Thursday 3:35-5:00pm Eastern Time
- **Location:** 34-101
- **Office Hour:** Thursday 5:00-6:00 pm Eastern Time, 38-344 Meeting Room
- **Discussion:** Piazza
- **Homework Submission:** Canvas
- **Contact:**
 - For external inquiries, personal matters, or emergencies, you can email us at *efficientml-staff [at] mit.edu*.
 - If you are interested in getting updates, please sign up [here](#) to join our mailing list to get notified!
- **Prerequisites:** 6.191 Computation Structures **and** 6.390 Intro to Machine Learning. Students who don't full-fill the prerequisites will be de-registered in the second week of class



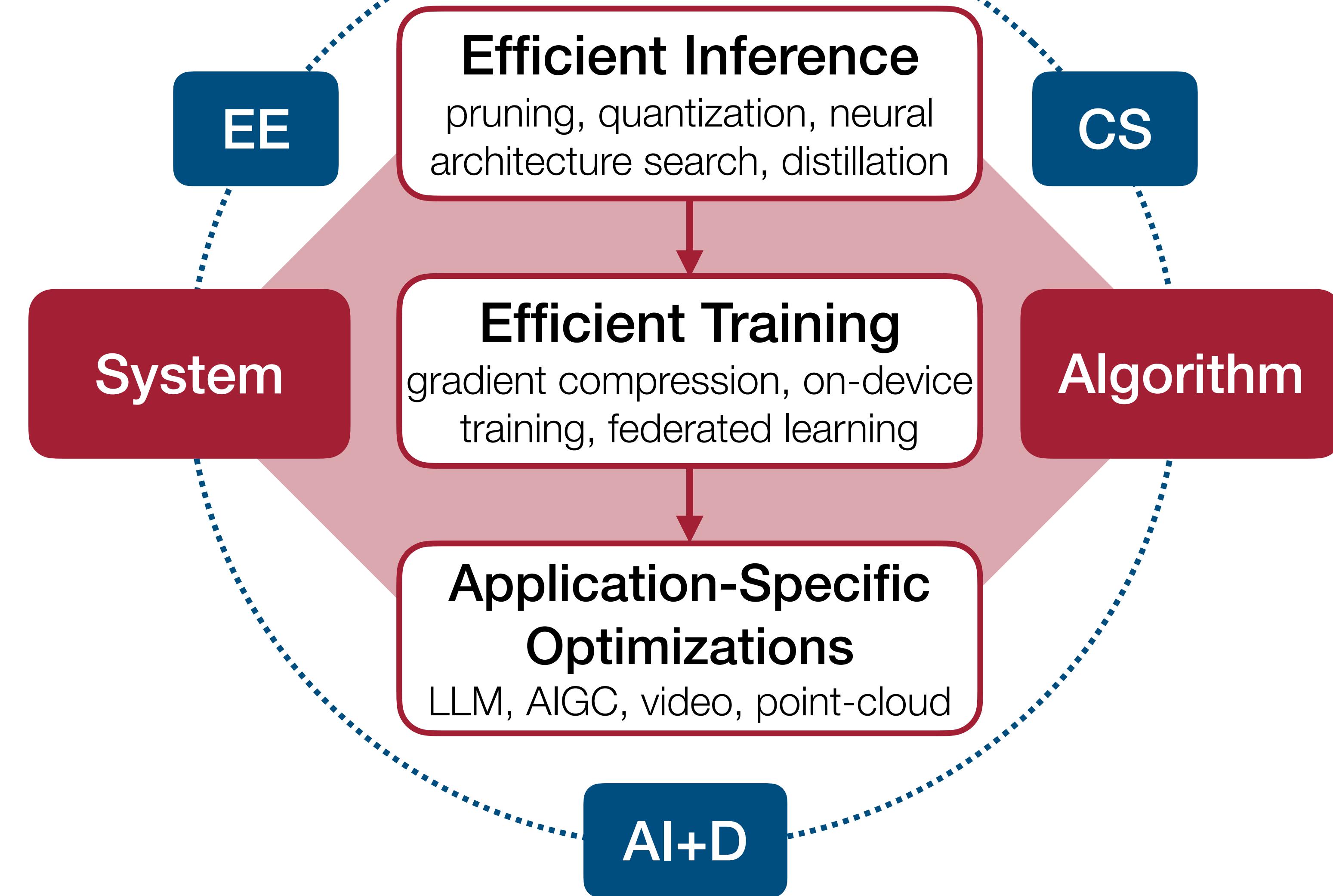
Course Overview



Course Overview



Course Overview



Course Overview

Computation Structures

(6.1910[6.004]), pre-req

Hardware Architecture for Deep Learning
(6.5930[6.825])

Microcomputer Project Lab
(6.2060[6.115])

EE

System

Efficient Inference

pruning, quantization, neural architecture search, distillation

CS

Algorithm

Efficient Training
gradient compression, on-device training, federated learning

Application-Specific Optimizations
LLM, AIGC, video, point-cloud

AI+D

Introduction to Machine Learning (6.3900[6.036]), pre-req

Deep Learning (6.S898)

Advances in Computer Vision (6.8300)

Computer System Architecture
(6.5900 [6.823])

Software Performance Engineering
(6.1060[6.172])

Mobile and Sensor Computing
(6.1820[6.808])

Labs

- There will be 5 labs over the course of the semester.
 - Lab1: Pruning
 - Lab2: Quantization
 - Lab3: Neural architecture search
 - Lab4: LLM compression
 - Lab5: LLM deployment on laptop
 - In Lab 5, we will deploy a quantized LLM model on your computer to run a chatbot. The minimum requirement is as follows:
 - Operating systems: MacOS, Linux, or Windows
 - Processor: x86 (Intel/AMD) or ARM (Apple M1/M2)
 - Memory: 8 GB
 - Available storage: 5 GB

Assignments and Grading

- **Grading**
 - 5 Labs ($15\% \times 5$)
 - Final Project (25%), group of 3-4
 - Proposal (5%)
 - Presentation + Final Report (20%)
 - Participation Bonus (4%)
- All assignments are due by **11:59PM on the due date** (submitted online)
- **Late Policy for Labs**
 - You will be allowed **6 total homework late days without penalty for the entire semester.**
 - The allowed late days are counted by day (i.e., each new late day starts at 11:59 pm ET).
 - Once the allowed late days are exceeded, the penalty is 50% per late day.
 - The homework is worth zero credit 2 days after exceeding the late day limit.

Course Overview

Prerequisites

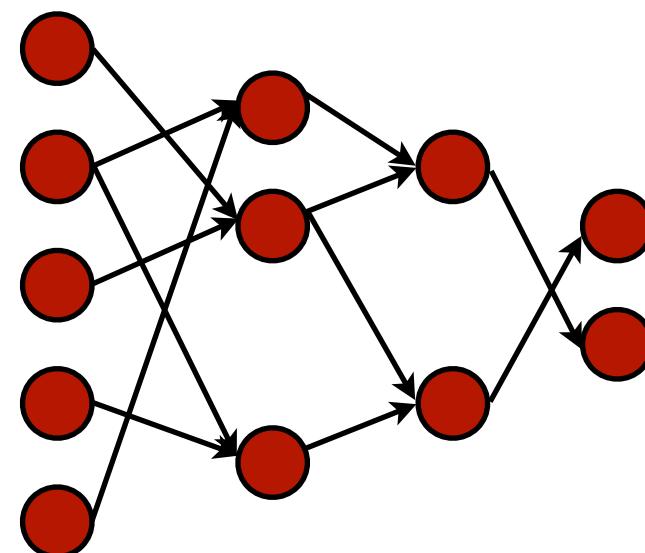
- 6.191 Computation Structures and 6.390 Intro to Machine Learning.
- Students who don't full-fill the prerequisites will be de-registered in the second week of class.
- If you believe you have equivalent prior experience (e.g., a computer architecture course taken during your undergraduate studies at another institution), you may petition for consideration. Please submit your **Petition Form** by **this Friday, 11:59:59 PM EST**.



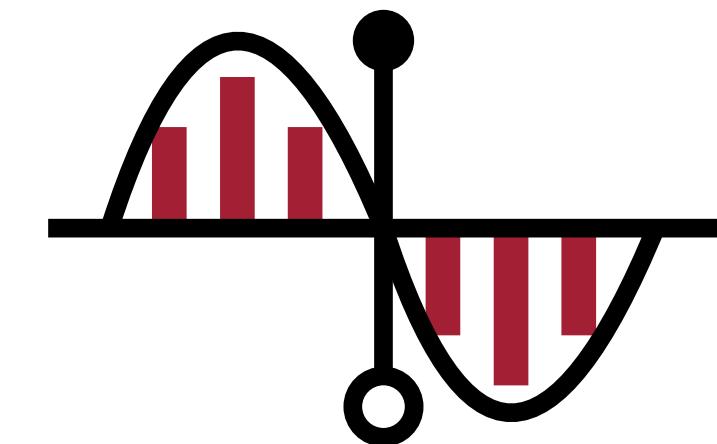
Labs



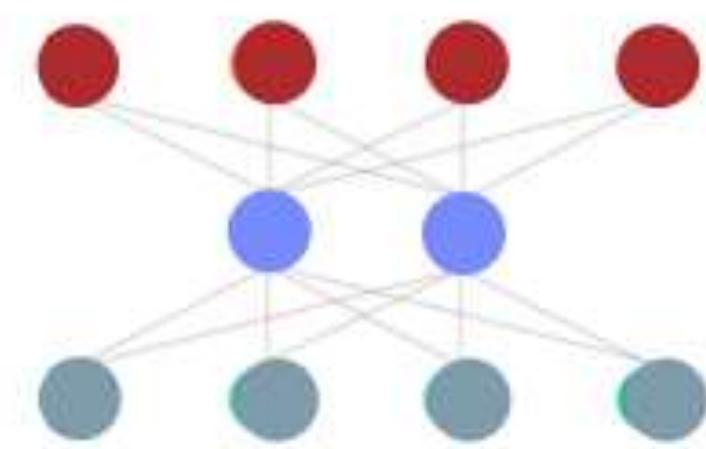
Lab 0 – Getting Started with PyTorch



Lab 1 – Pruning



Lab 2 – Quantization



Lab 3 – Neural Architecture Search

Lab 4 – LLM Compression



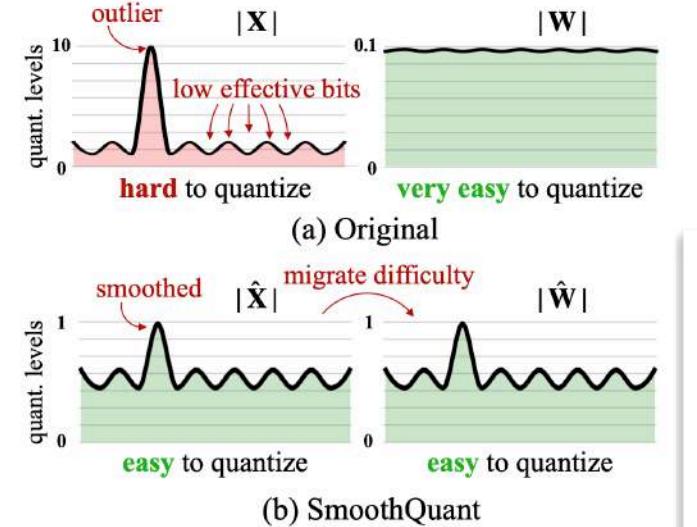
Lab 5 – LLM Deployment on Laptop

Final Projects

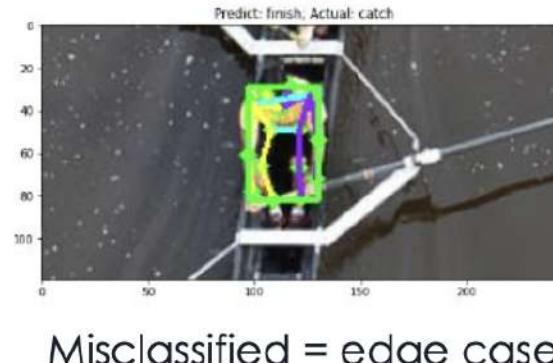
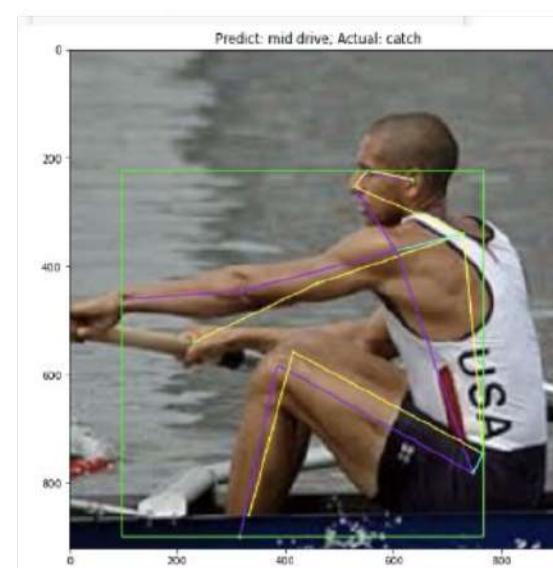
The poster features a blue background with a white rowing boat icon. The title "aiP: AI-Row" is at the top right. Below it is a network diagram with nodes and connections. Text includes: "The First Real Time Mobile ML Analytics for both Para and Non-Para Rowers", "Emelie, Sol, Veronica", and "MIT 6.S965: TinyML and Efficient Deep Learning Computing <https://efficientml.ai>". A small text at the bottom says "5th in the 2022 World Rowing Championships".

SmoothQuant: Accurate and Efficient Post-Training Quantization for LLMs

| | LLM (100B+) Accuracy | Hardware Efficiency |
|---------------------|-------------------------|------------------------|
| ZeroQuant | ✗ | ✓ |
| Outlier Suppression | ✗ | ✓ |
| LLM.int8() | ✓ | ✗ |
| SmoothQuant | ✓ | ✓ |



- We propose SmoothQuant, an **accurate** and **efficient** post-training-quantization (PTQ) method to enable 8-bit weight, 8-bit activation (**W8A8**) quantization for LLMs.
- Since **weights are easy** to quantize while **activations are not**, SmoothQuant smooths the activation outliers by **migrating the quantization difficulty from activations to weights** with a mathematically equivalent transform!



Misclassified = edge cases

Knowledge Distillation and Quantization for Efficient Keyword Spotting 6.S95

Arman Dave and Julian Hamelberg

Optimizing TinyEngine Kernels and Experimental Kernel Generation with Multistaging and Meta-Programming

...

Anne Ouyang, Pranav Krishna

NoiseOut

Real-time Speech Enhancement on Mobile phones

Presenter: Team NoiseOut: {William Zhao, Mark Jabbour, Ian Lee}



Takeaways

- Know the key efficiency metrics for deep learning computing
- Be familiar with how to speed up inference and training of neural networks on resource-constraint platforms
- Understand the tradeoffs between different optimization techniques
- Be aware of recent research trends and industry practice
- Get hands-on experience compressing and implementing LLM, having LLM deployed on a laptop