Carnegie Mellon University

18-349 Guest Lecture:

Machine Learning

Tianshu Huang

(1) When should I use ML / DL?
(2) How do I develop and deploy ML to the edge?

This lecture brought to you by:

RFML

Interested in machine learning research and working with **real systems**, not just canned datasets?

> **Embedded HW/SW** development for learning-enabled radar applications
> **Machine learning** for radar and radar+X fusion

Contact: tianshu2@andrew.cmu.edu

This lecture brought to you by:

RFML: 18-484B

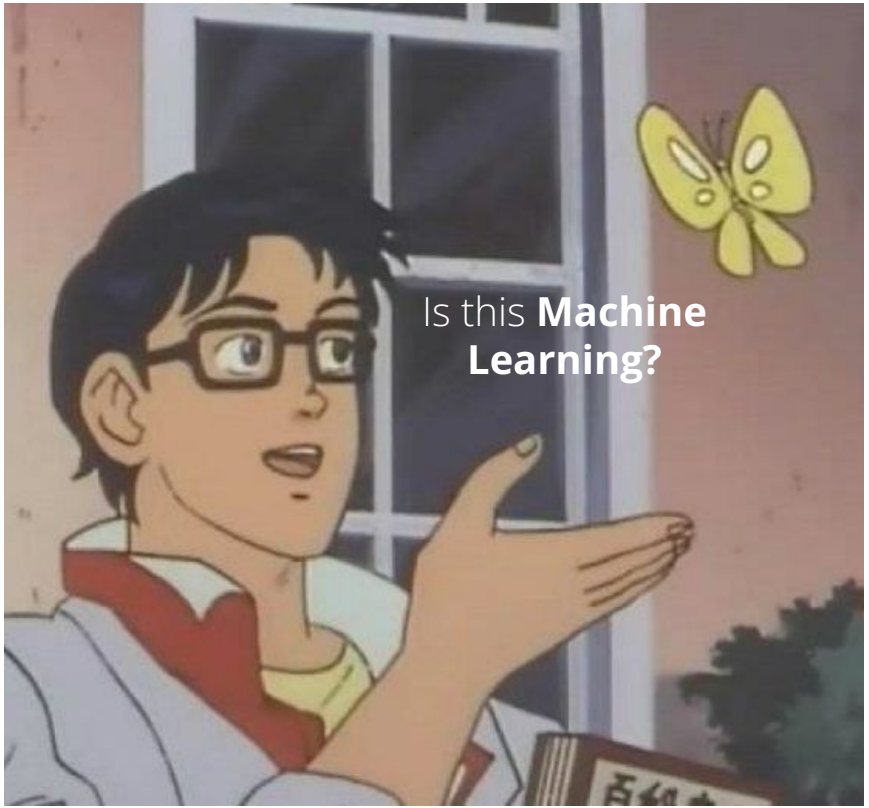
Special Topics in Embedded Systems: Radio Frequency Machine Learning

Learning-based approaches have revolutionized how we approach sensing and perception. The computer vision community, in particular, has embraced machine learning as a core paradigm of the field, driving rapid advances in the state of the art in both computer vision and machine learning. However, to date, there has been relatively little overlap between the wireless sensing and machine learning communities.

This research-focused course will provide students with Machine Learning **or** Wireless Sensing background the knowledge and skills needed to effectively engage in cross-disciplinary collaborations which push the frontiers of Machine Learning for Wireless Sensing. The course will feature traditional lectures covering key background, as well as research lectures covering the current state-of-the art and guest lectures from academic researchers and industry insiders. Students will gain experience working with wireless systems such as radars and spectrum analyzers, as well as large datasets collected from these systems. Students will also complete a research project applying Machine Learning techniques to a wireless sensing modality of their choice

“AI”

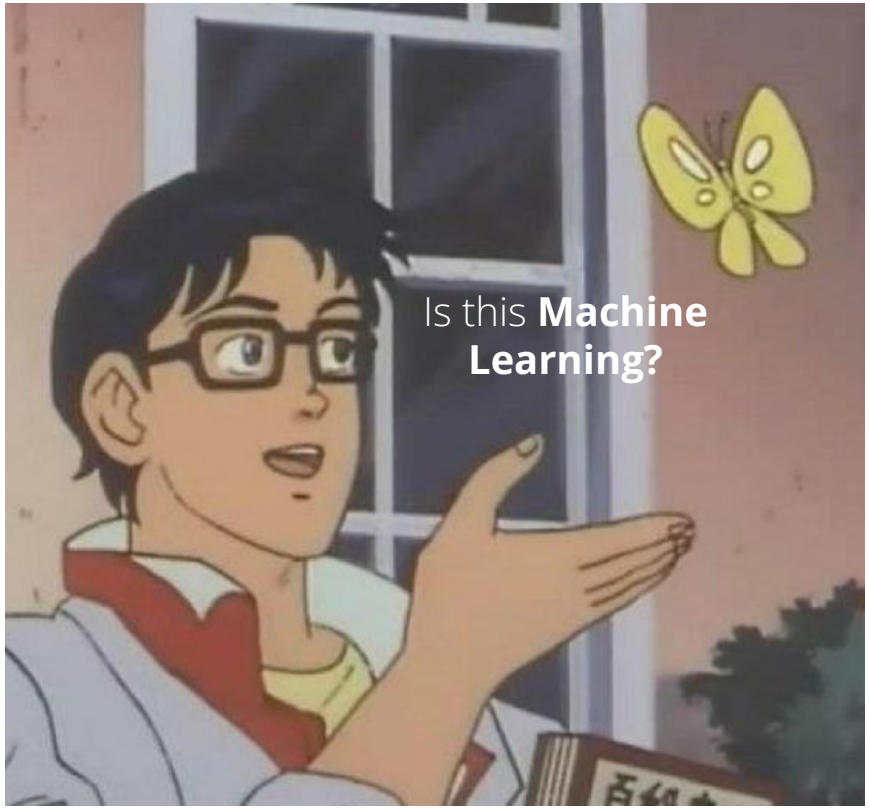
Machine Learning



Machine Learning

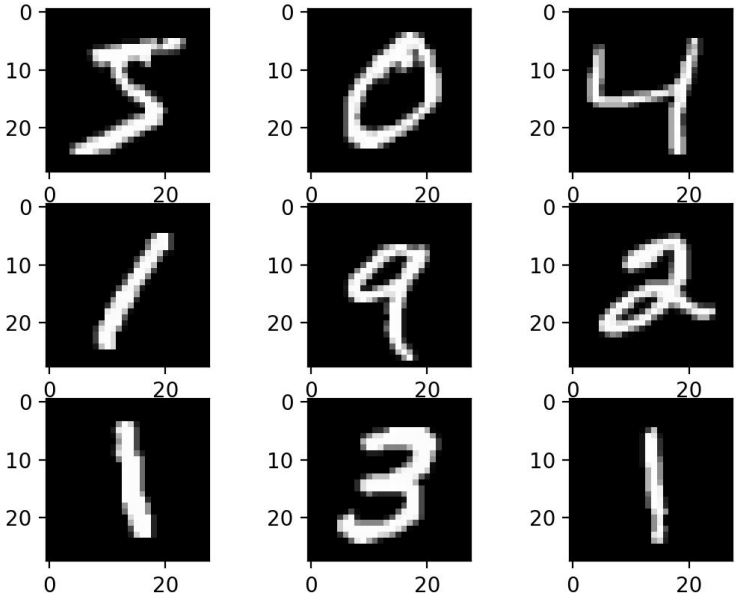
Empirical risk minimization:

1. Data
2. Risk function (e.g. accuracy, loss)
3. What model minimizes the risk function on the data?

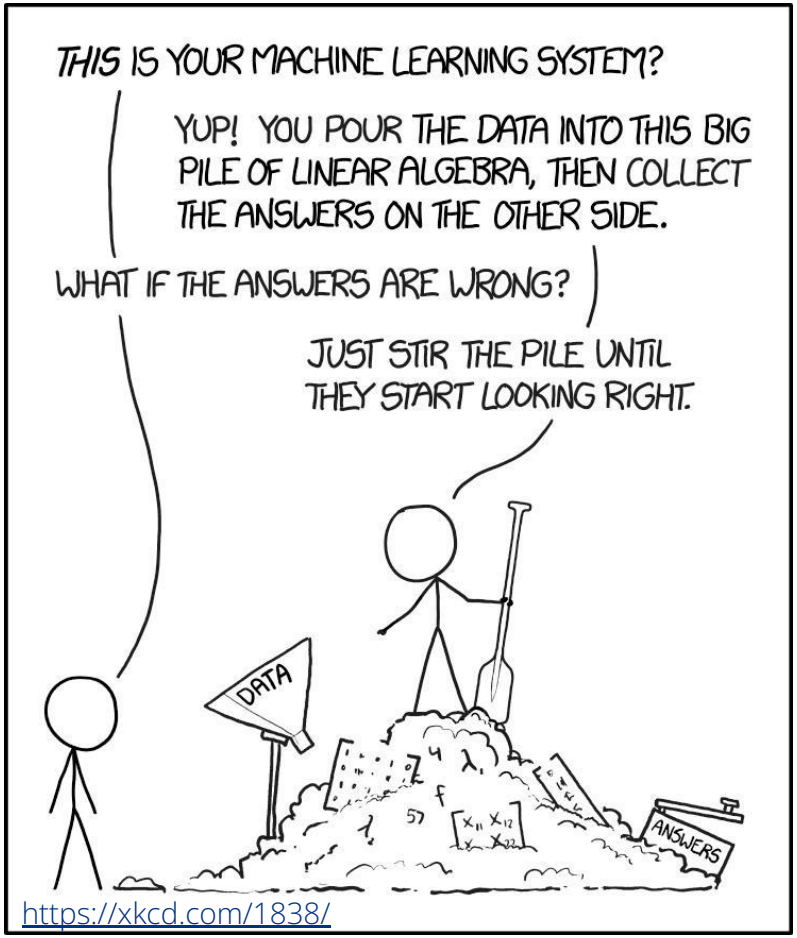


Machine Learning

Example: digit classification



Machine Learning



https://xkcd.com/1838/

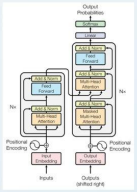
Deep Learning

STOP DOING DEEP LEARNING

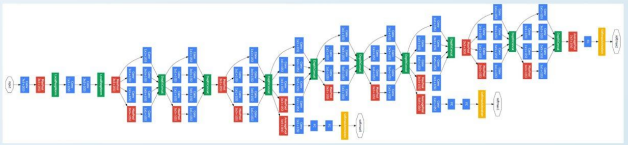
- PERCEPTRONS WERE ONLY EVER MEANT TO BE FULLY CONNECTED
- THOUSANDS OF PAPERS yet NO REAL-WORLD USE FOUND for going deeper than ONE LAYER
- Wanted to add more nonlinearity anyway for a laugh? We had a tool for that: It was called “KERNEL METHODS”
- “Yes please give me a network that can PAY ATTENTION TO ITSELF. Please give me PRETRAINED WEIGHTS for my YOLO-9000” - Statements dreamed up by the utterly Deranged

LOOK at what “Research Scientists” have been demanding your Respect for all this time, with the statistical methods & optimization algorithms we built for them

(This is REAL "Deep Learning”, done by REAL “ML Engineers”):



?????



??????????????????

“Hello I would like to learn **1.6 TRILLION** parameters please”

They have played us for absolute fools

https://www.reddit.com/r/okbuddyphd/comments/n2m6vz/stop_doing_deep_learning/

Deep Learning

The Higher Mysteries

NOoO!! You have to understand Deep learning!!!

Merging Models modulo Permutation Symmetries (Ainsworth et al., 2022)
Neural Networks are Decision Trees (Aytekin, 2022)
Universal Approximation Theorem
...

No one understands deep learning

Practitioners
Large scale learning

I don't understand deep learning

The Natural Image Manifold

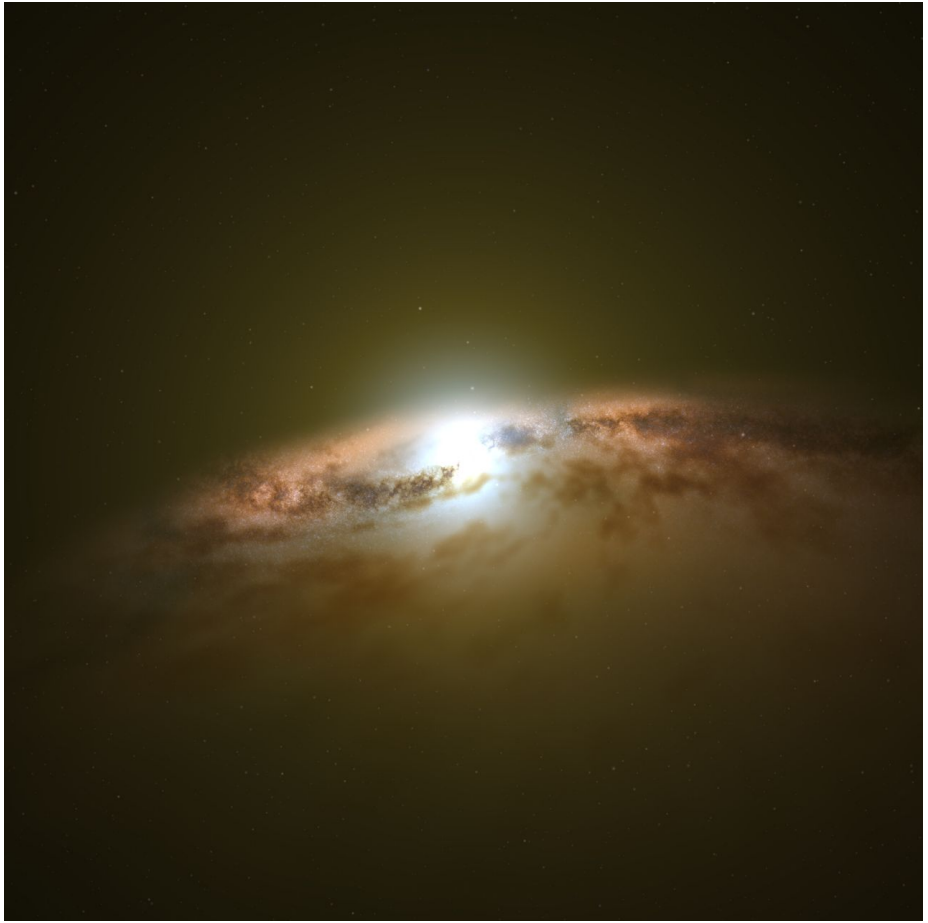
The **natural image manifold** is a surface embedded in a **high dimensional space** that is:

(1) **low-dimensional**,
(2) **highly nonlinear**, and
(3) **locally smooth**.

The Natural Image Manifold

The **natural image manifold** is a surface embedded in a **high dimensional space** that is:

(1) **low-dimensional**,
(2) **highly nonlinear**, and
(3) **locally smooth**.



The Natural Image Manifold

The **natural image manifold** is a surface embedded in a **high dimensional space** that is:

(1) **low-dimensional**,
(2) **highly nonlinear**, and
(3) **locally smooth**.

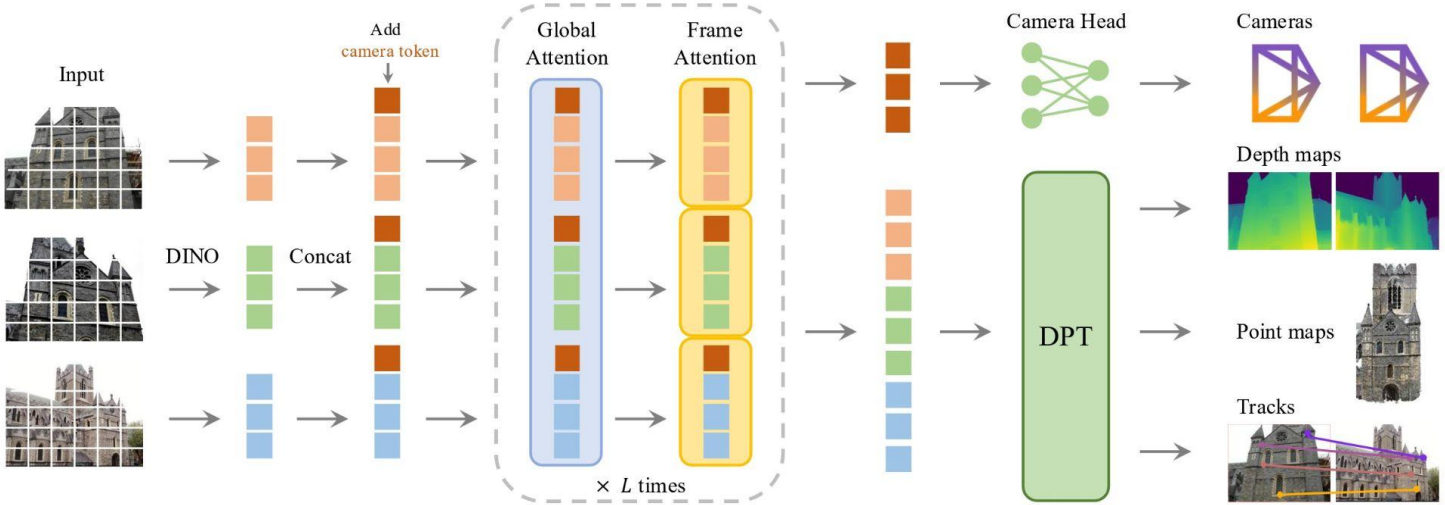
Practical Deep Learning

See **Adversarial Robustness**.
Image from "Robust Physical-World Attacks on Deep Learning Models," Eykholt et al, 2018.

Pre-Trained Models & Transfer Learning

[https://www.reddit.com/r/machinelearningmemes/comments/g1mxz8/transfer_learning_101/]

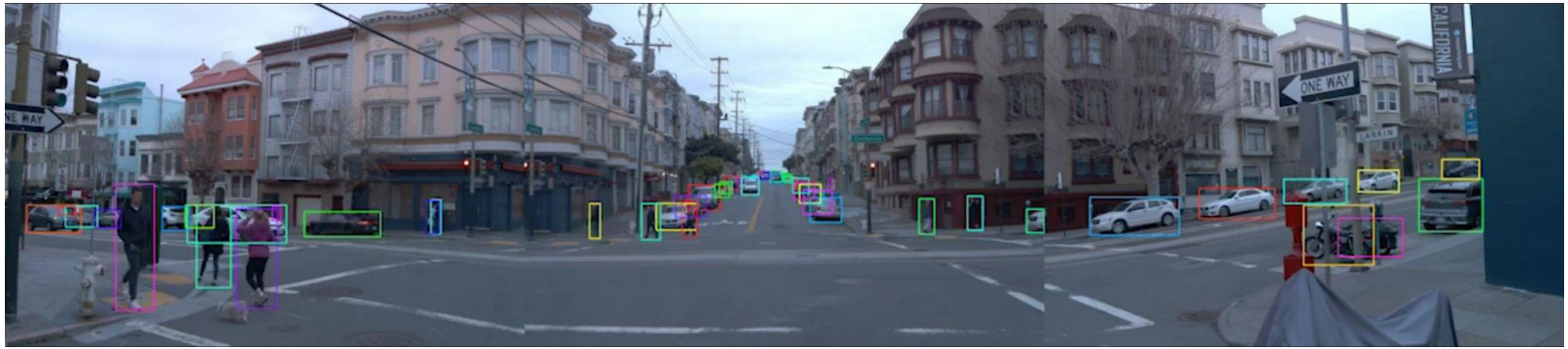
Pre-Trained Models & Transfer Learning



VGGT: Visual Geometry Grounded Transformer, Meta AI, 2025

Practical Deep Learning

- Train from scratch?
- Transfer learn?
- Fully pre-trained model?



Deep Learning on the Edge

Please inspect your power cables if you own a RTX 4090, even if you don't use it for deep learning.

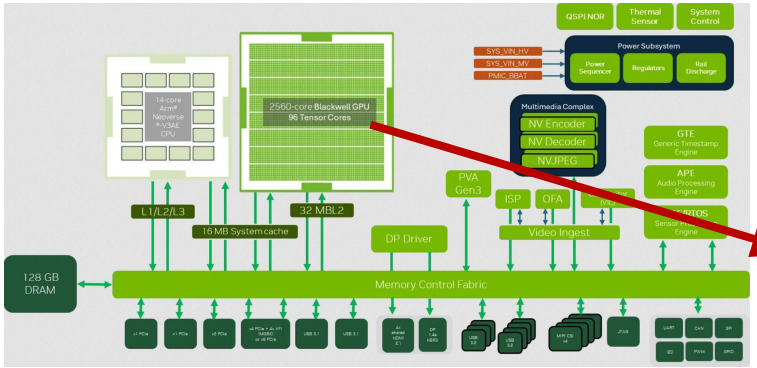
Edge Accelerators (NPU)

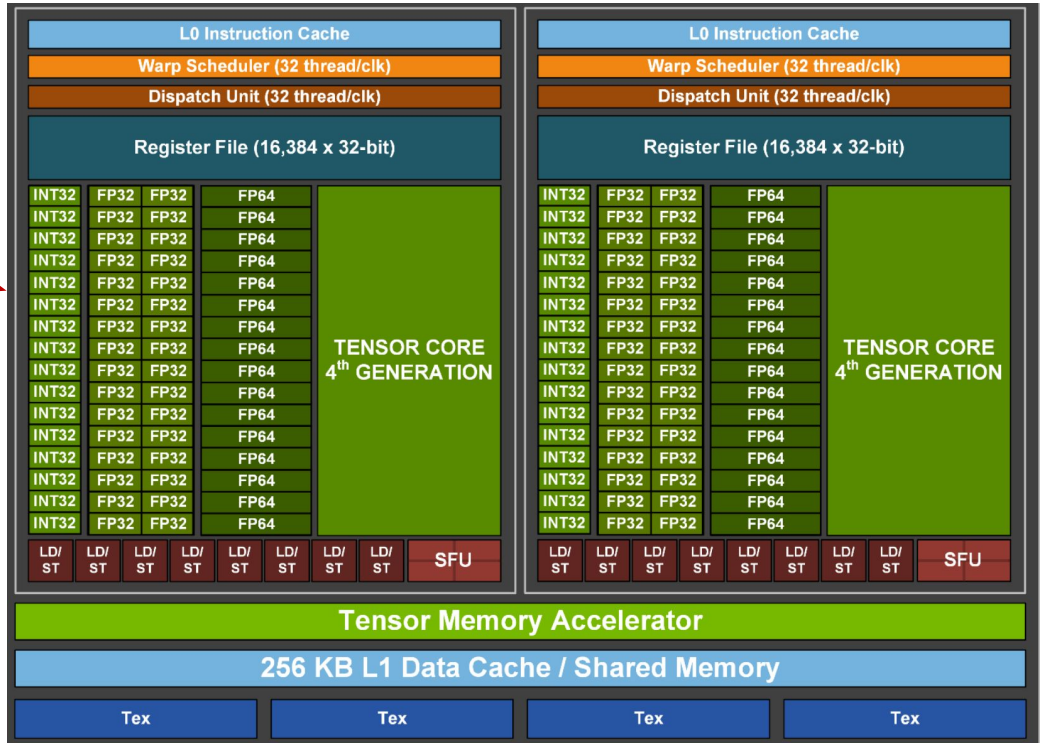
Key design feature:

- 1 instruction = matrix/tensor multiplication with many flops
  - Jetson/CUDA: ~1k
  - Hexagon/VLIW DSP: ~10k per word
  - Coral/TPU: >>100k
- Optimize data flow within each instruction/word

Edge Accelerators (NPU)

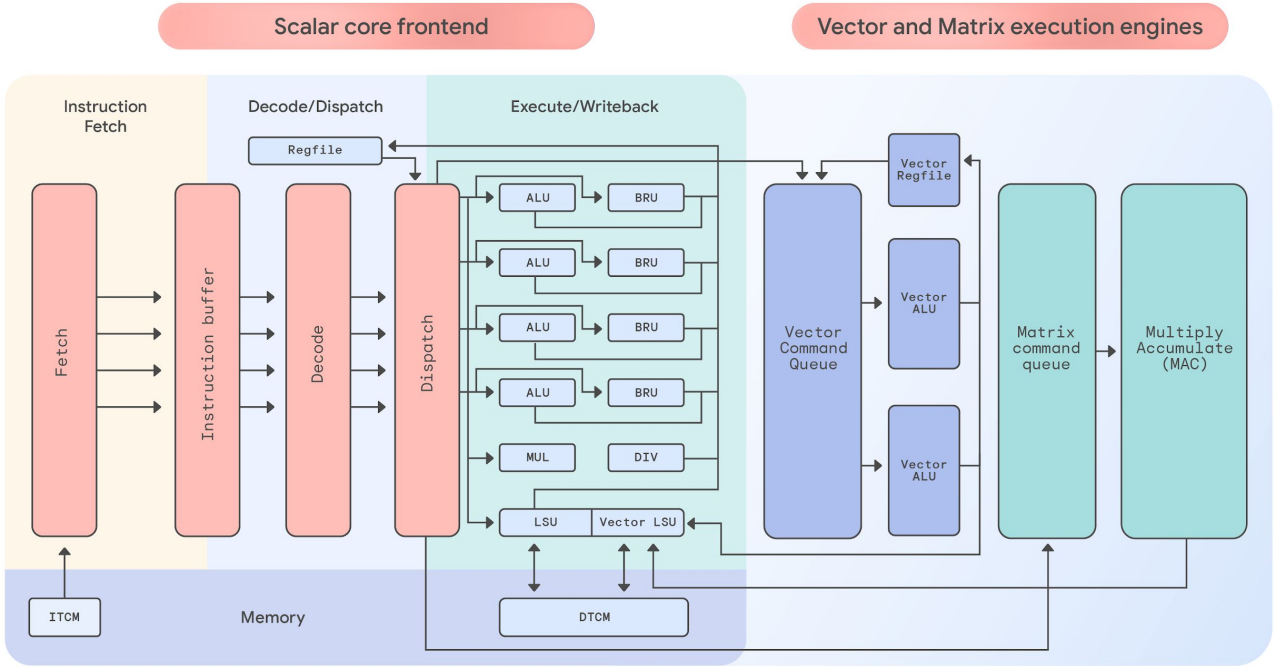
Nvidia Jetson (GPU Lineage)



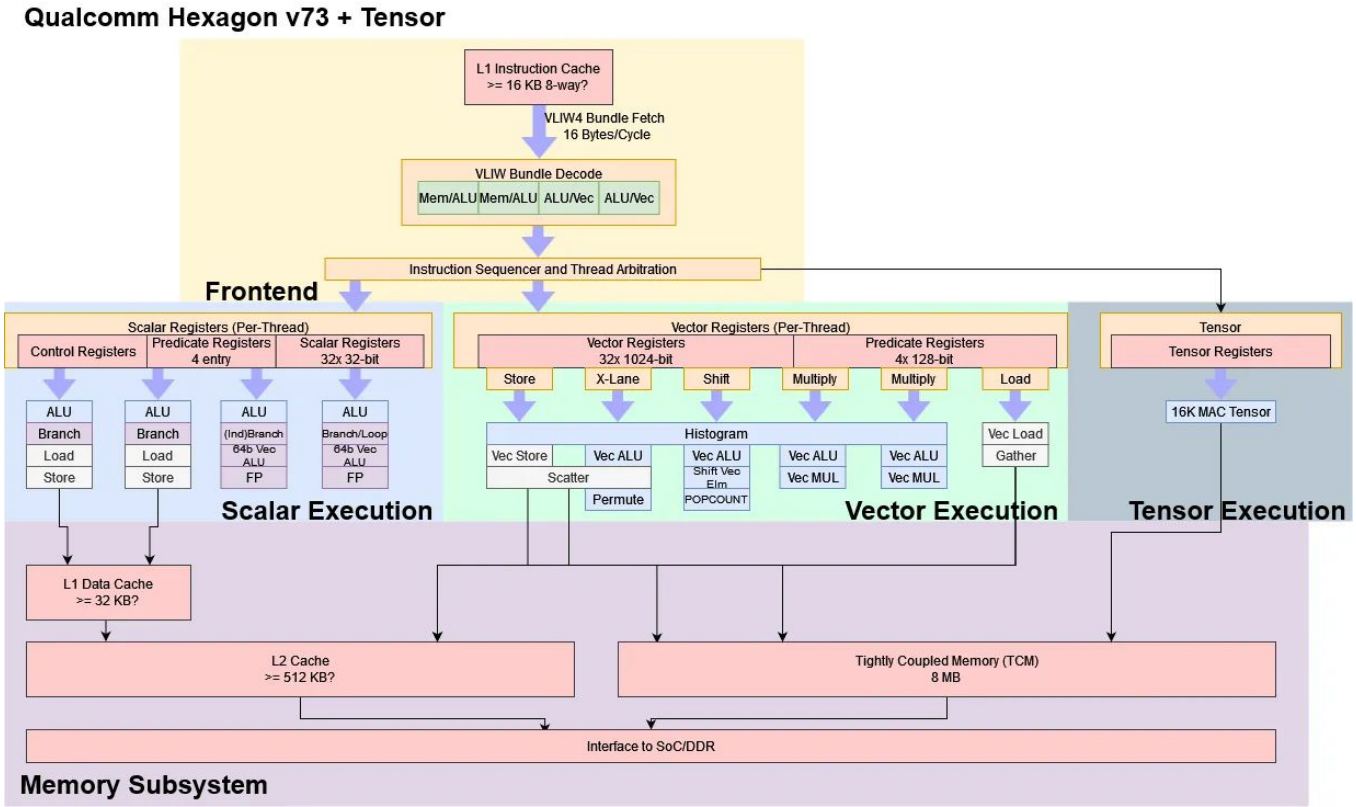


Edge Accelerators (NPU)

Google Coral (TPU Lineage)



Edge Accelerators (NPU)



Pros/Cons

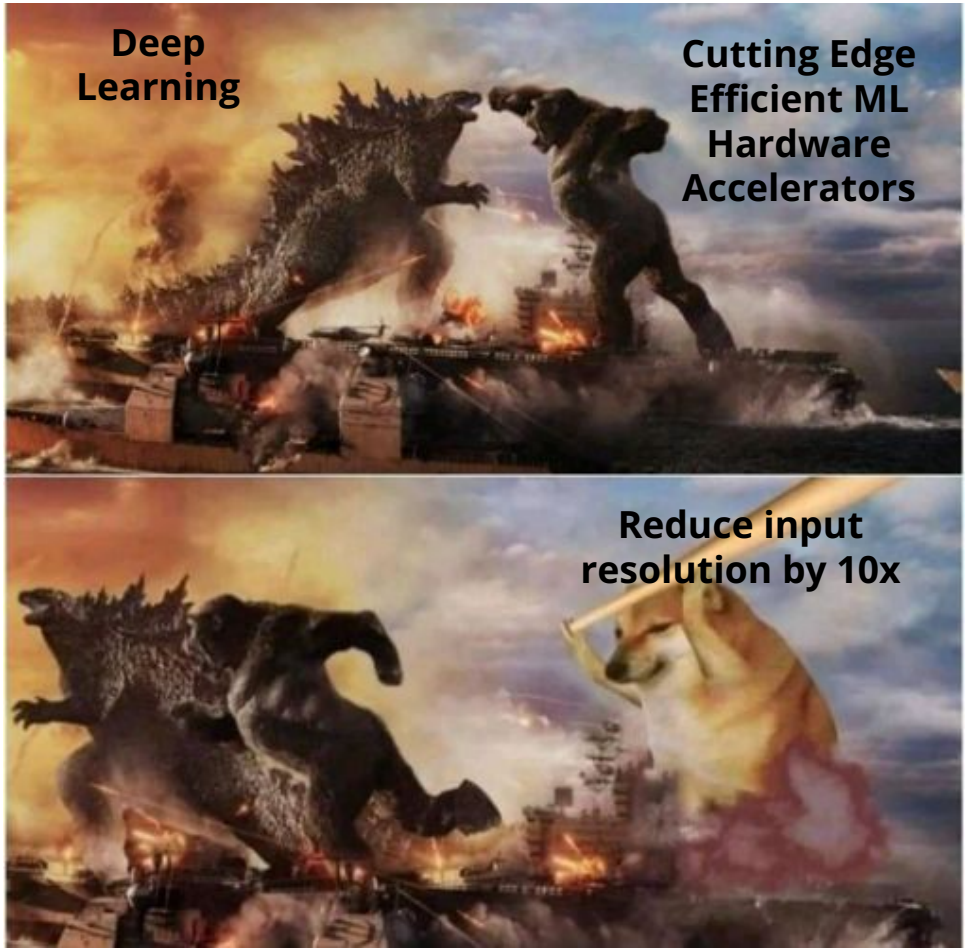
uArch	Description	Pros	Cons
Jetson (GPU)	Many SIMT units which can do medium-size vector or matmul ops		
Hexagon (DSP)	Single VLIW unit with large vector/matmul ops		
Coral (TPU)	Single execution unit with huge tensor/matmul ops		

Deep Learning on the Edge

- Trained a neural network with a given architecture
- Weights are ~~32~~16-bit float
- Fixed power budget

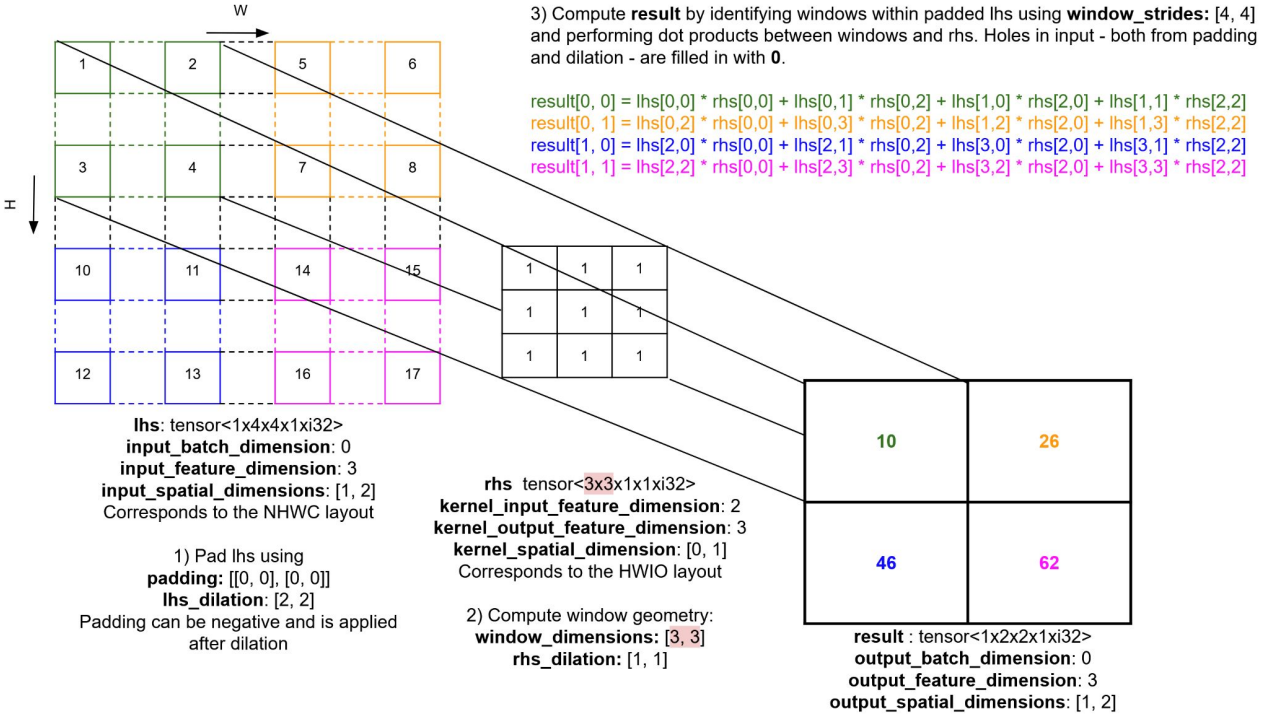
Simplest is Best

- Cloud offloading
- Reduce input resolution
- Don’t use deep learning



Use better software

- Compilers, e.g. XLA
- Custom CUDA / accelerator code

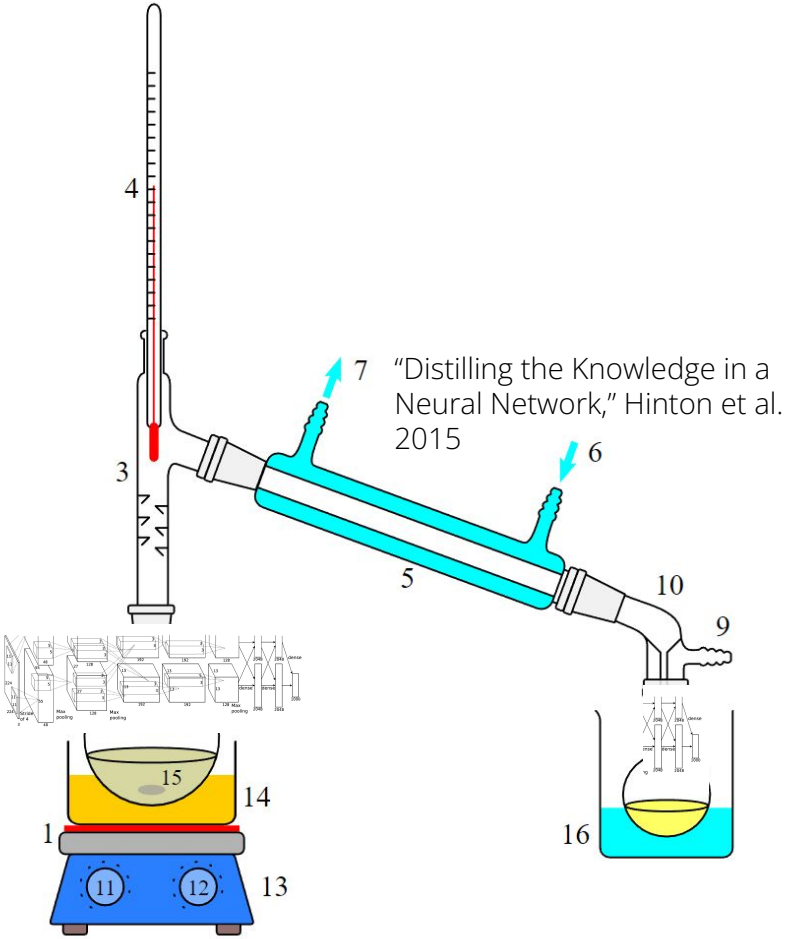


Quantization

Memory bandwidth dominates*

- low rank, k-means approximations
- float32 → float16
- even better: int8
- even even better: int4
- binary, ternary is even possible!

Distillation & Co-Design



"Distilling the Knowledge in a Neural Network," Hinton et al. 2015

TL;DR

1. Don’t use machine learning
2. Don’t use deep learning
3. Don’t use deep learning on the edge (do it in the cloud)
4. Don’t train deep learning models (use a pre-trained model optimized for edge deployment)
5. Don’t train deep learning from scratch (use transfer learning)
6. Give up (or take a machine learning class)