Next class & Proposal extension

Project Feedback & Lab 2:

- Profs and TAs will be in class to give feedback on project ideas
- Lab 2: Replicate Lab 1 on your project data & baseline model
- Project proposal due date extended 1 week to next Friday **October 6**.

Lecture recordings

- If you (plan to) miss a lecture, use this form to ask for the recording:

http://bit.ly/11767-lecture-rec-req

Carnegie Mellon University
Language Technologies Institute

Lecture 6: Compression I

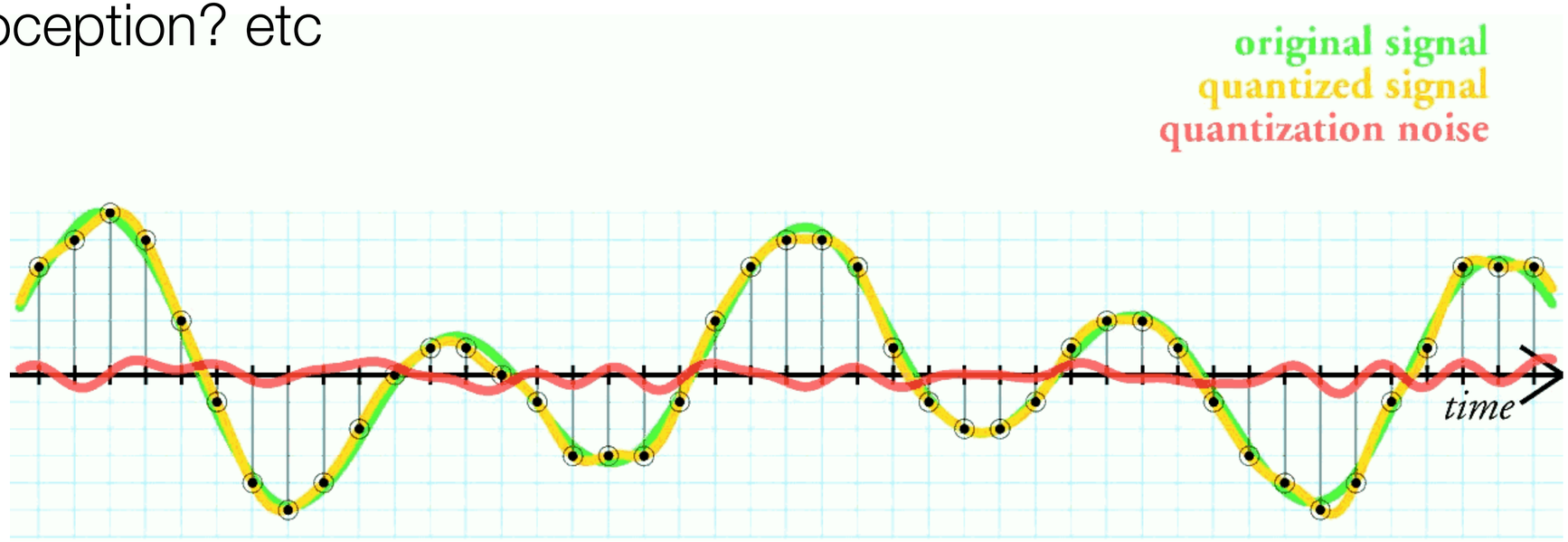
Quantization

Emma Strubell & Yonatan Bisk

What is quantization?

The world is analog

- How many bits are in an “image”?
- How any bits are in a sound wave?
- How many bits is your brain sending to move your arm? Experience proprioception? etc



Wikipedia Quantization

What is compression?

Top 10 detections?

15MB

5MB

1.4MB

1512 x 2016 pixels

What is compression?

1.4MB

756x1008 pixels

400KB

378x504 pixels

100KB

189 x 252 pixels

What is compression?

100KB

189 x 252 pixels

38KB

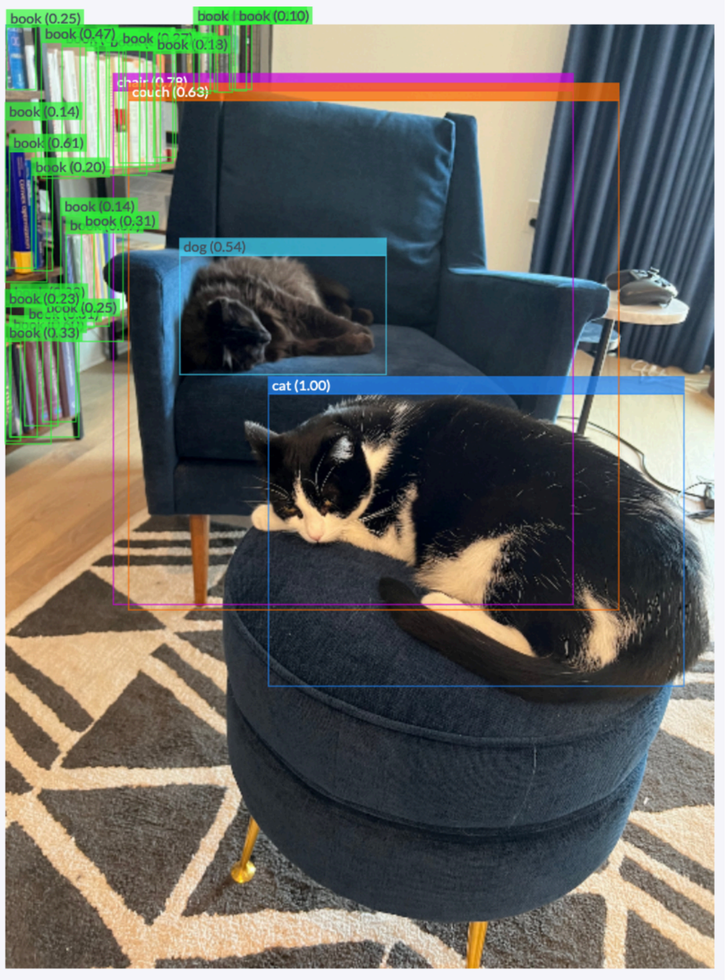
95x126 pixels

16KB

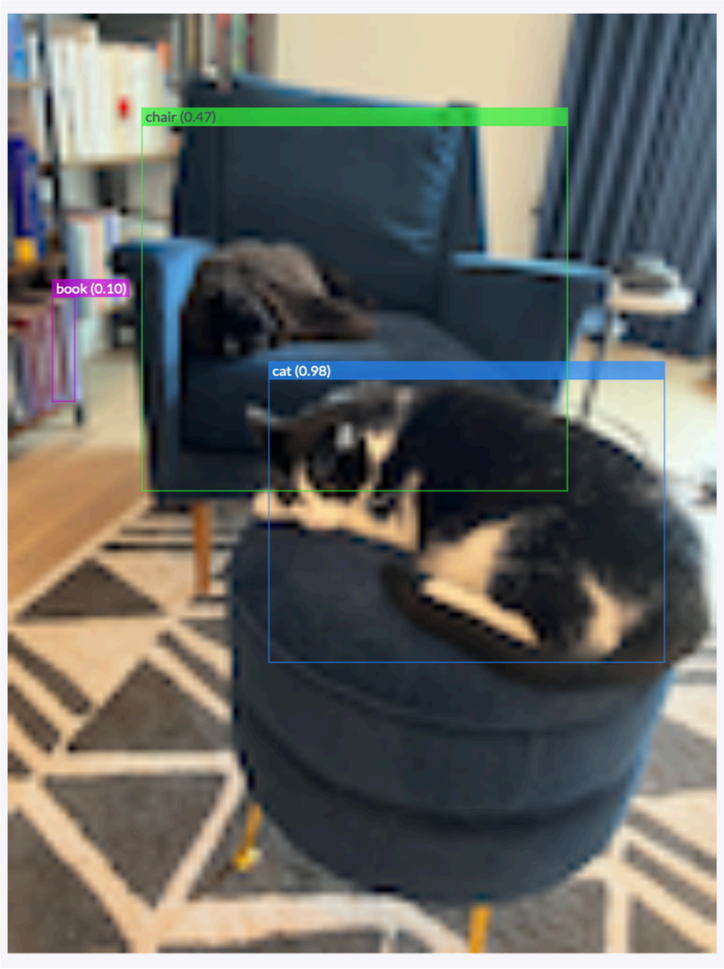
48x63 pixels

What is compression?

When does compression affect meaning?

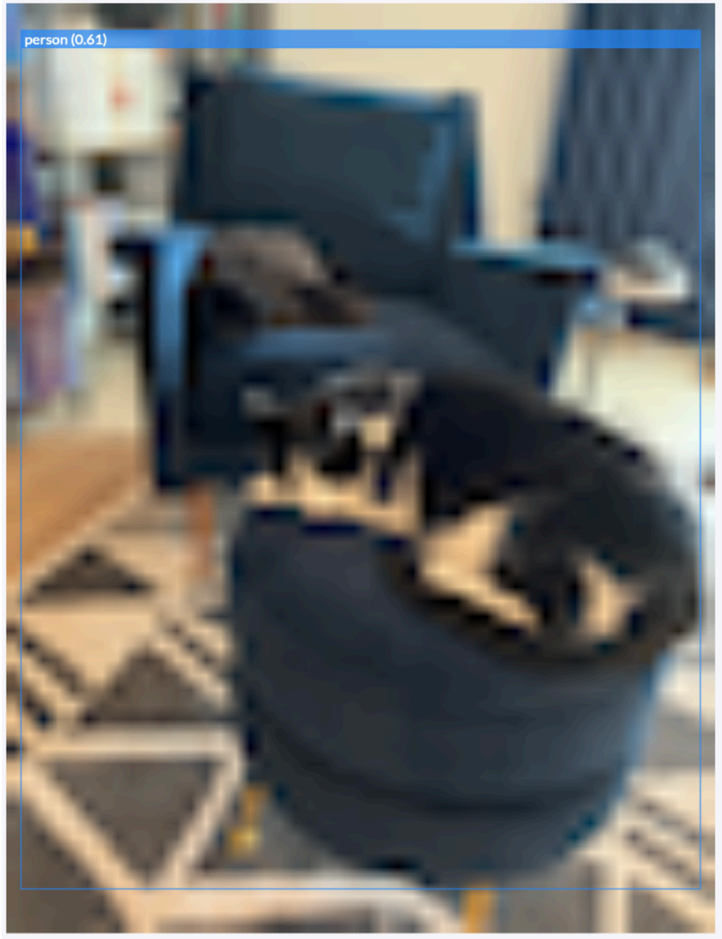


5MB



38KB

95x126 pixels



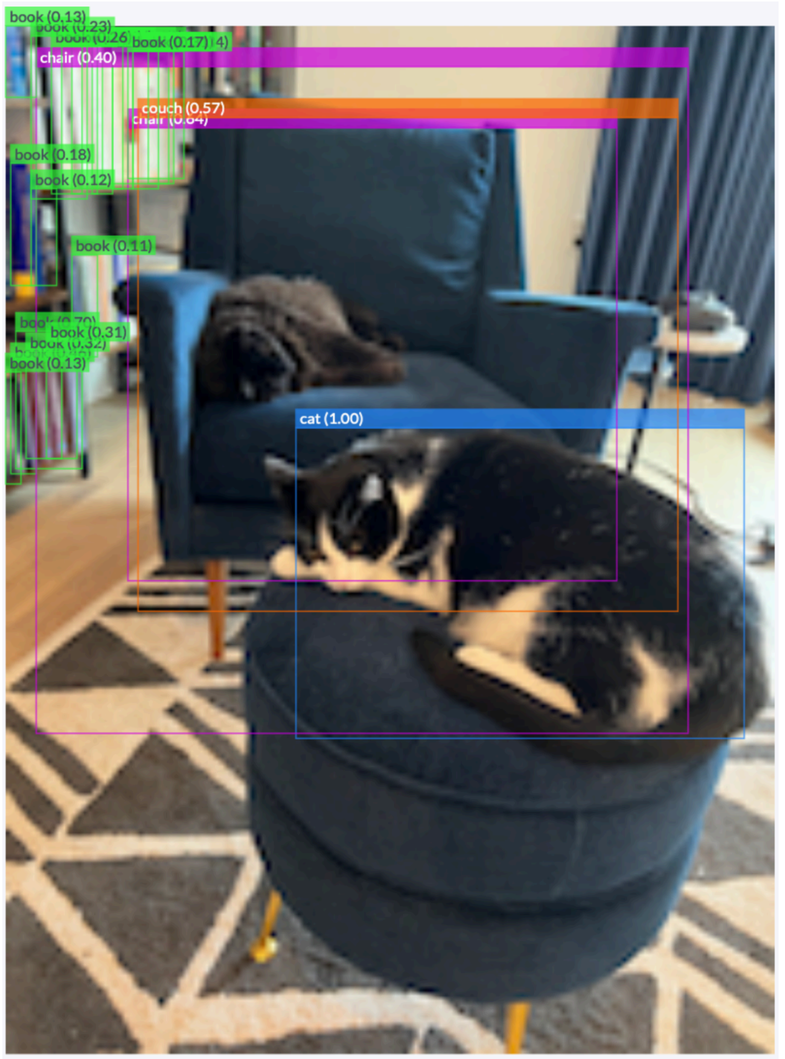
16KB

48x63 pixels

Not all bits are created equal

PNG

JPG

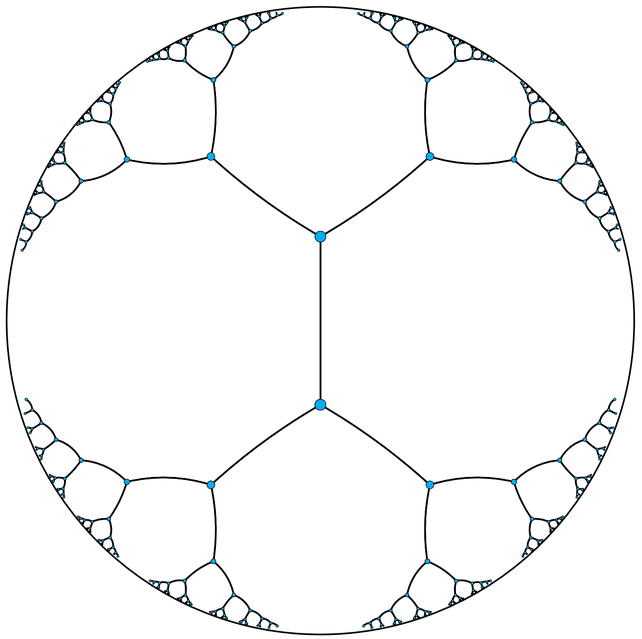


Information vs Bits

Example: Poincare Embeddings

- Euclidean vs Hyperbolic Spaces
- representational power != representation

$$d(\boldsymbol{u}, \boldsymbol{v}) = \operatorname{arcosh}\left(1 + 2\frac{\|\boldsymbol{u} - \boldsymbol{v}\|^2}{(1 - \|\boldsymbol{u}\|^2)(1 - \|\boldsymbol{v}\|^2)}\right)$$



		Reconstruction
		10
ASTROPH	**Euclidean**	0.376
N=18,772; E=198,110	**Poincaré**	0.703
CONDMAT	**Euclidean**	0.356
N=23,133; E=93,497	**Poincaré**	0.799
GRQC	**Euclidean**	0.522
N=5,242; E=14,496	**Poincaré**	0.990
HEPPH	**Euclidean**	0.434
N=12,008; E=118,521	**Poincaré**	0.811

Poincaré Embeddings for Learning Hierarchical Representations

https://arxiv.org/abs/1705.08039

What is quantization in neural networks?

Key modeling decisions:

- Target bit width?
- Encoding (INT8, FP8(1-5-2), …)?
- Dynamic or static bit width?
- (Nonlinear) mapping to quantized space?
- Post-hoc quantization, or quantization-aware training (QAT)?

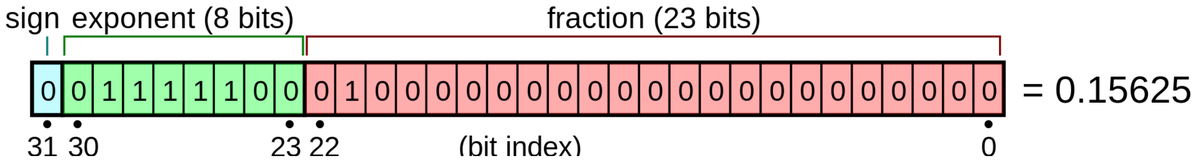


Table 1: Dynamic range comparison between proposed FP8 and other existing floating point formats.

Data Type	Bit Format (s, e, m)	Max Normal	Min Normal	Min Subnormal
IEEE-754 float	1, 8, 23	3.40e38	1.17e−38	1.40e−45
IEEE-754 half-float	1, 5, 10	65 535	6.10e−5	5.96e−8
FP8 (proposed)	1, 5, 2	57 344	6.10e−5	1.52e−5

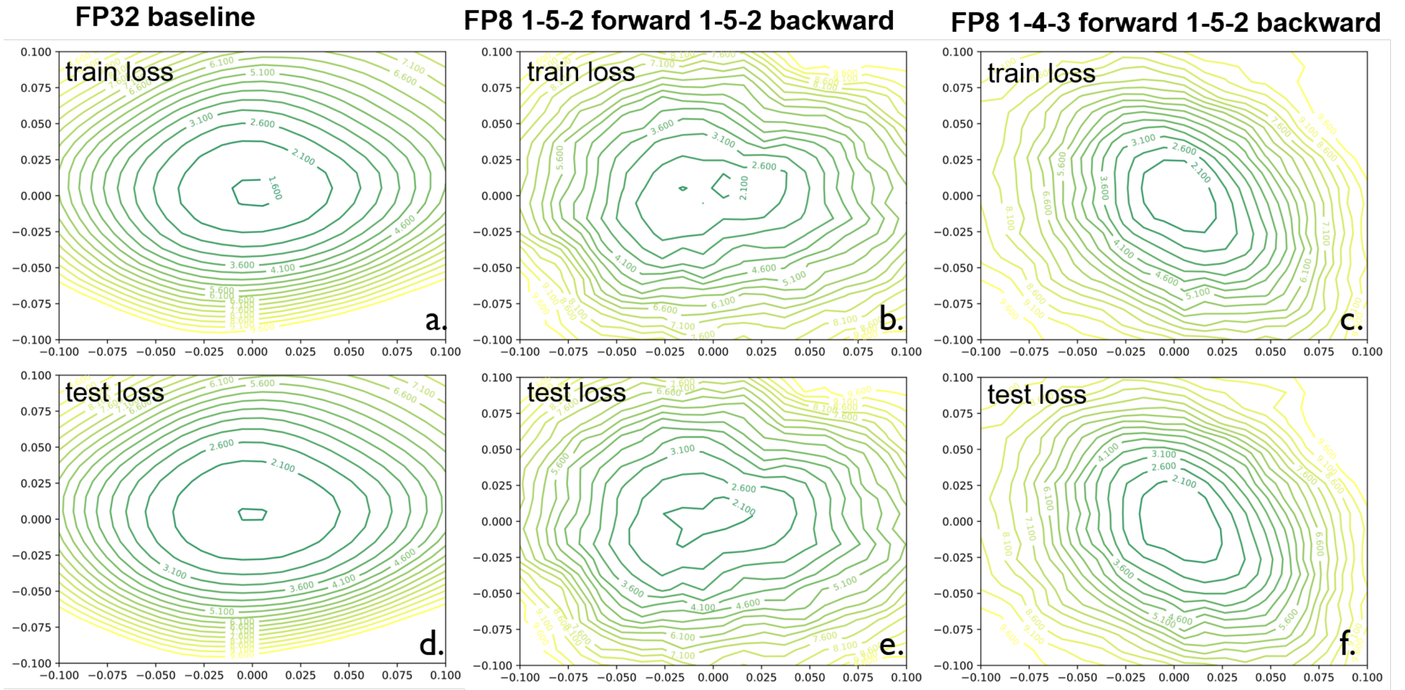


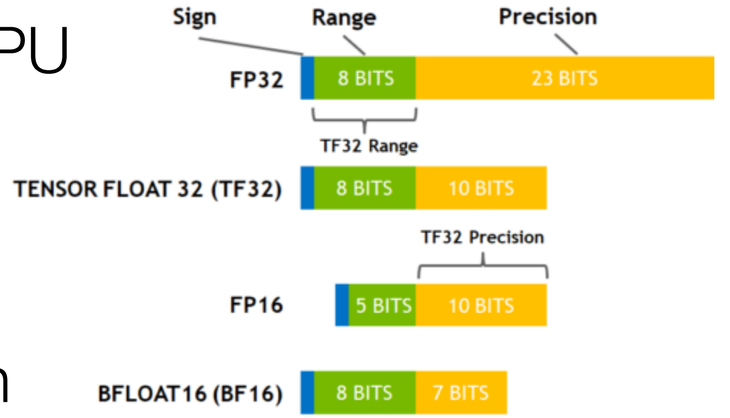
Table:
N. Mellempudi, et al. Mixed Precision Training With 8-bit Floating Point. arXiv 2019.

Figure: https://en.wikipedia.org/wiki/File:Float_example.svg

Figure: X. Sun, et al. Hybrid 8-bit Floating Point (HFP8) Training and Inference for Deep Neural Networks. NeurIPS 2019.

Mixed-precision training w/ 16 bits

- Implemented in hardware — Faster and more memory efficient!
  - float16 matmul up to **16x faster** than float32 on A100 GPU
  - Can double performance for memory-bound models.



- **Automated Mixed Precision (AMP):** Automatically set the precision of all floating point operations in an application by profiling error sources and types (Nathan et al. 2016).

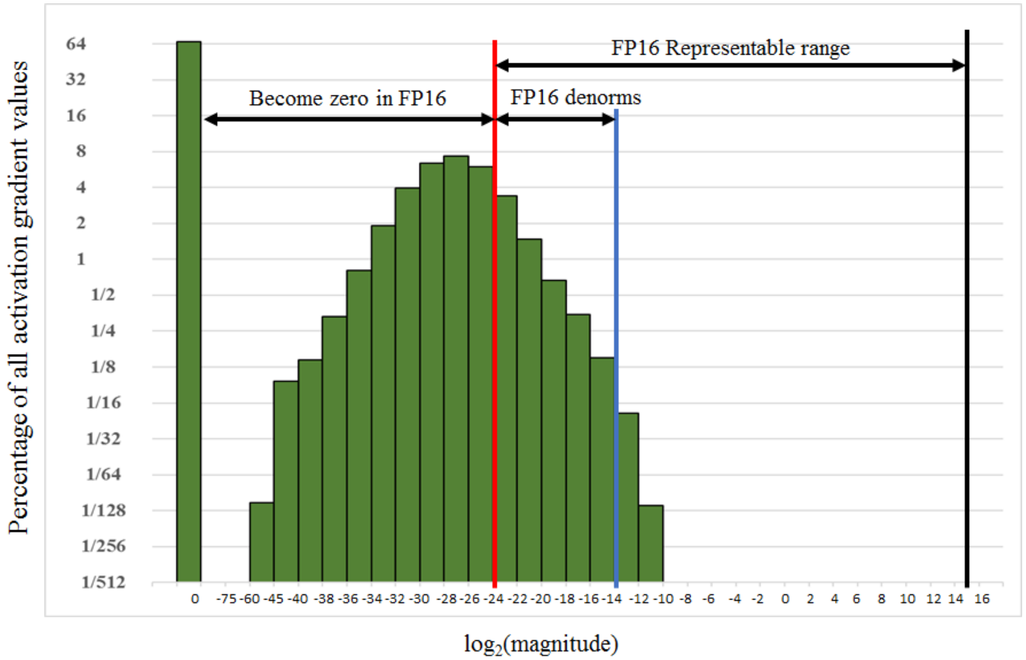
Numerical fault	Threshold for promotion in precision
Large round-off	> t_1% instances have > t_2% error ratio
Large diff in addend exponents	> t_3% instances have exponents differing by > t_4
Severe cancellation	Any instance has > t_5 most signif. bits cancelled
Near overflow/underflow	Any instance has result magnitude > t_6 or < t_7

Figures: https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html

https://dev-discuss.pytorch.org/t/more-in-depth-details-of-floating-point-precision/654

Mixed-precision training w/ 16 bits

- For NN training: **Loss scaling** to preserve small gradients (Micikevicius et al. 2018).
- Store weights, activations, gradients in float16. Accumulate updates in float32.
- Multiply by a scaling factor to shift gradients into representable range.



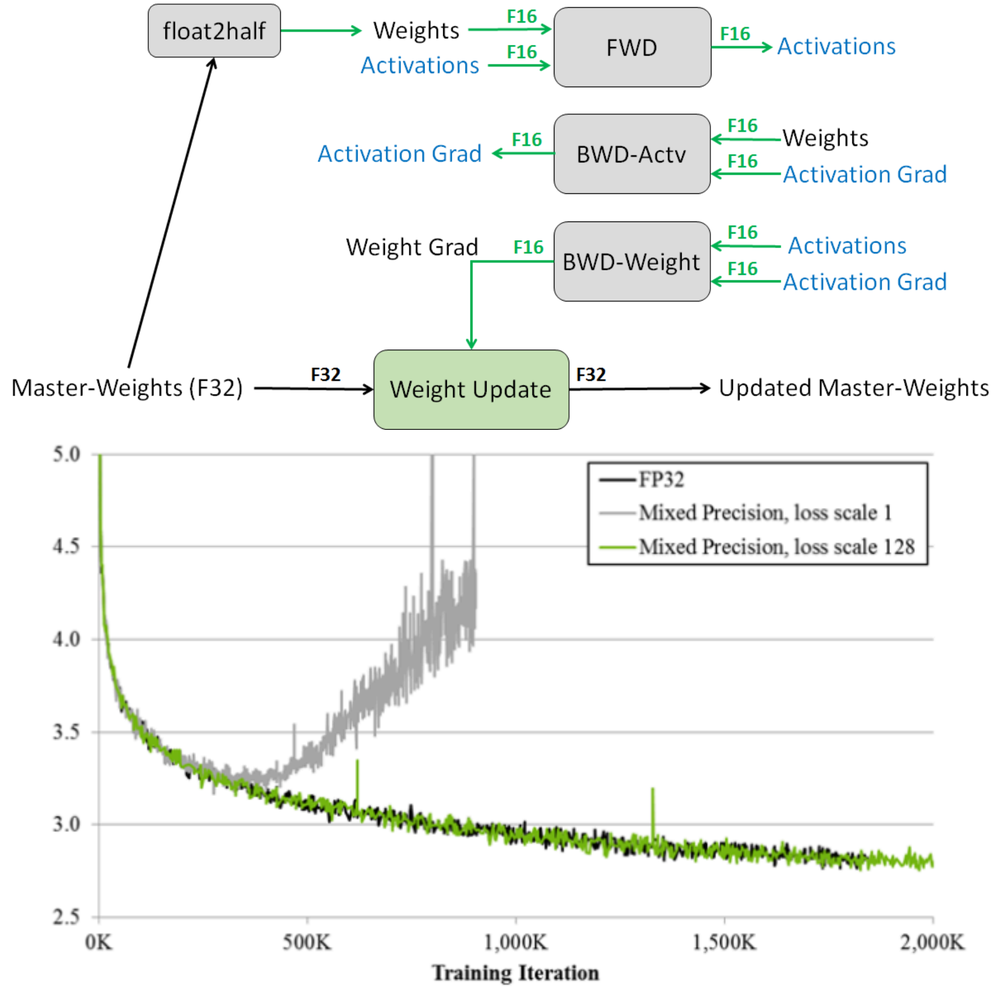


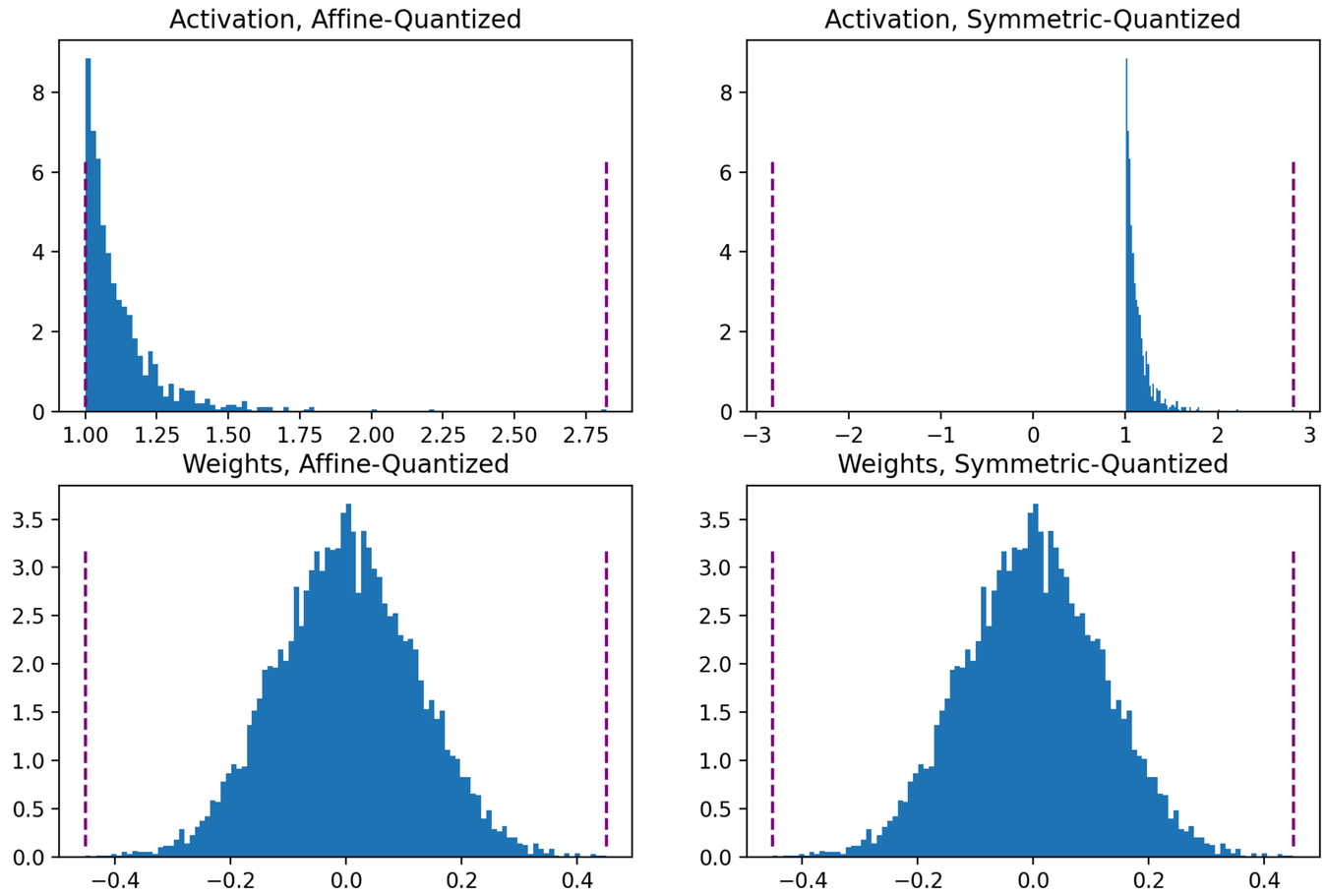
Figure: https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html

Post-training quantization (PTQ)

- You have a large model, and want to improve inference efficiency/space: Quantize an existing model using **post-training quantization (PTQ)**.
- Mapping function: How to transform full-precision values to quantized space?
  - **Linear mapping:** $Q(r) = round(\frac{r}{S} + Z)$ **inverse:** $\tilde{r} = (Q(r) - Z) \cdot S$
  - ***r*** is the input, ***S*** is the scaling factor and ***Z*** is the zero point.
  - ***S*** is ratio of input range [α, β] to output range [α_q, β_q]: $S = \frac{\beta - \alpha}{\beta_q - \alpha_q}$
  - ***Z*** is bias mapping zero between input and output: $Z = -(\frac{\alpha}{S} - \alpha_q)$
  - Note that $\tilde{r} \neq r$ and the difference is the **quantization error**.
- Choose parameters via **calibration** on data.
  - **Affine:** α=min(r), β=max(r); **Symmetric:** -α=β=max(|max(r)|, |min(r)|); center at 0.

Post-training quantization (PTQ)

- **Affine:** α=min(r), β=max(r); **Symmetric:** -α=β=max(|max(r)|, |min(r)|); center at 0.



Post-training quantization (PTQ)

Dynamic / weights-only PTQ:

- Just model weights are pre-quantized; activations quantized on-the-fly during inference, with scaling factors computed dynamically for each input and layer.
- Good for LSTMs and Transformers where weights memory I/O can be a bottleneck.
- Calibrating and quantizing adds computational overhead.

Static PTQ:

- Model weights are pre-quantized, and clipping range is pre-calibrated and fixed using a validation set.
- Fast: removes computational cost of converting float↔int.
- Sensitive to distribution shift — may require regular re-calibration.

https://pytorch.org/blog/quantization-in-practice/

Quantization-aware training (QAT)

- **Quantization-aware training (QAT):**
  Learn model parameters that minimize quantization error. $Q(r) = round(\frac{r}{S} + Z)$
  - Simulate $Q(r)$ followed by $\tilde{r}$ during training, include quantization error in loss fn.
- **Problem:** *round* is not differentiable!
  - **Straight-through estimator** (STE; Bengio et al. 2013): Set incoming gradients to threshold fn equal to outgoing gradients; Disregard gradient (or lack thereof) of round fn itself.
  - As an alternative, can also introduce **additive noise** to simulate quantization noise (Défossez et al. 2021).

$$z_{q,\text{train}} = \text{clamp}(z, m - L \cdot \sigma, m + L \cdot \sigma) + L \cdot \sigma \cdot \frac{\mathcal{U}[-1, 1]}{2^B}$$

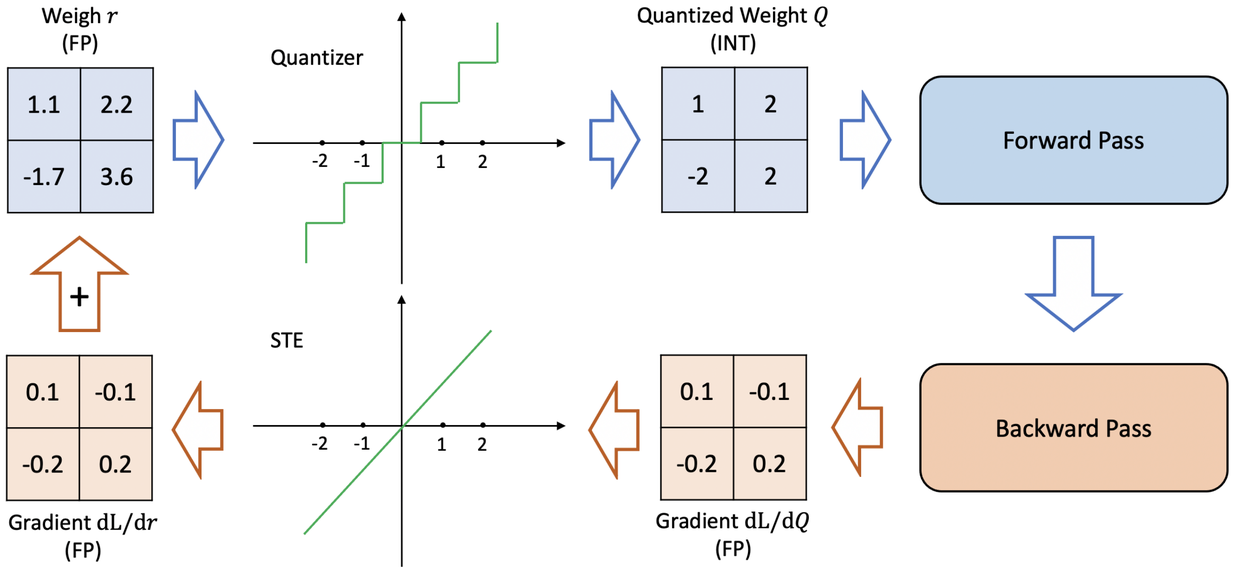


Figure: A. Gholami, et al. A Survey of Quantization Methods for Efficient Neural Network Inference. 2021.

Is quantization real?

- Much quantization in ML research is “fake” — need hardware/library support for fast mixed-precision, low-precision operations.
- PyTorch, TensorFlow both support basic fake quantization for QAT.

FPGAs

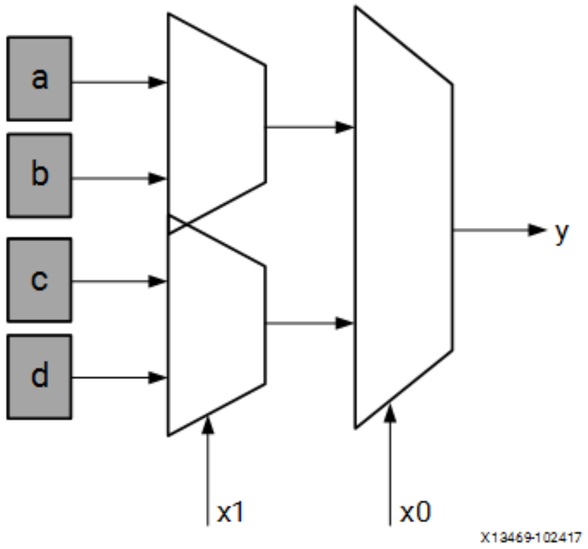


TABLE II. COMPARISON WITH FIXED-POINT CNN AND BNN ACCELERATORS

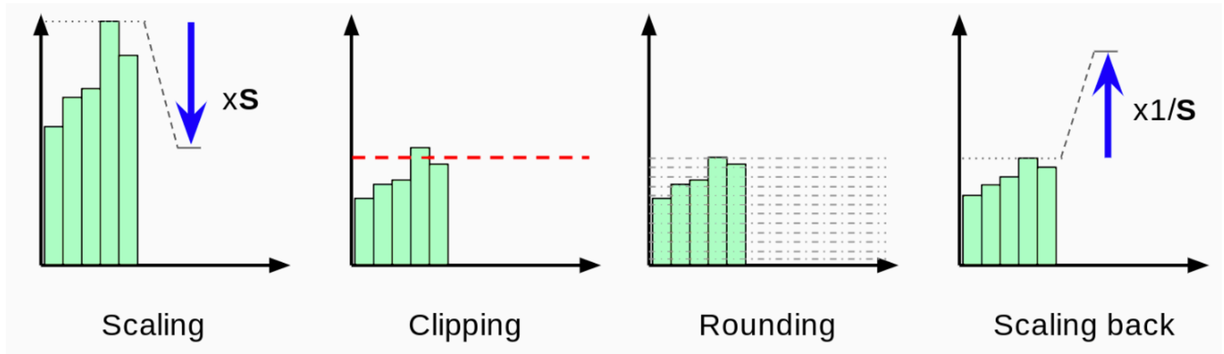
Accel-erator	Platform	kLUTs	DSP	BRAM	Throughput (FPS)	Power (W)	Performance (GOPS)	R-Efficiency (GOPS/kLUT)	P-Efficiency (GOPS/W)	Accuracy	Bit-width	Dataset	CNN Size (GOP)
[3]	Stratix-V GSD8	120	760	1480	3.8	19.1	117.8	0.98	6.17	87.48%	8-16b	ImageNet	30.9
[13]	Zynq XC7Z045	182.6	780	486	4.45	9.6	136.97	0.75	14.22	86.66%	16b	ImageNet	30.76
[4]	Virtex-7 VC709	273.8	2144	956.5	391	30.2	565.94	2.07	22.15	83%	16b	ImageNet	1.45
[12]	Zynq XC7Z020	46.9	3	94	168.4[a]	4.7	207.8	4.43	44.2	88.54[b]	1-2b	CIFAR-10	1.22
[11]	Zynq XC7Z045	46.2	-	186	21.9k[a]	11.7	2465.5	54.1	210.5	80.1%[b]	1b	CIFAR-10	0.31
This work	Zynq XC7Z020	44	89	105.5	106	2.26	410.22	8.47	181.51	73.1%	2b	ImageNet	3.87

[a.] Accelerators [12] and [11] target at the CIFAR-10 dataset, which has an image size of 32 × 32.

[b.] The original BNN model obtains 50.42% top-5 accuracy on ImageNet [8], which declines greatly compared to our 73.1% accuracy.

Table: L. Jiao et al. Accelerating low bit-width convolutional neural networks with embedded FPGA. FPL 2017.

When is low so low it's high?

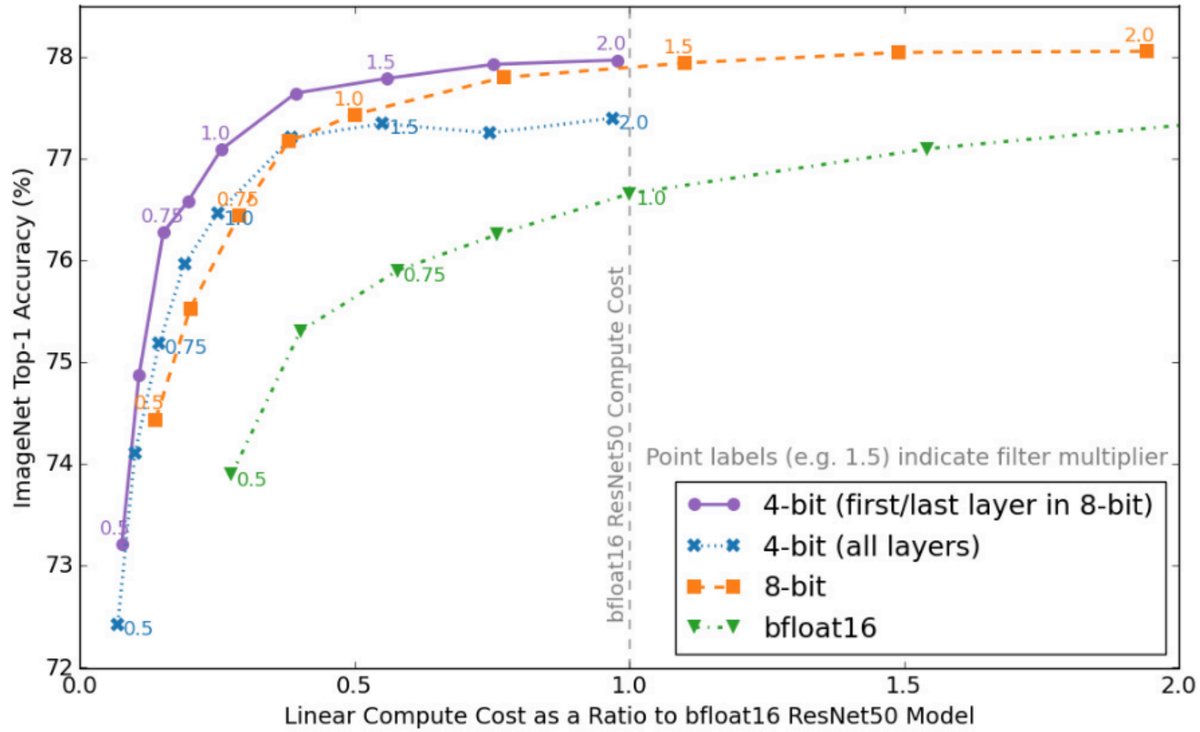


Input types	*TOPS*
float16	312
bfloat16	312
int8	624
int4	1248
binary	4992

NVIDIA A100 performance on quantized types.

Hardware multiplication of two n-bit numbers requires reducing n^2 bits to $2n$ bits i.e.,

$$a \cdot b = \sum_{0 \le i,j < n} 2^{i+j} \text{ AND } (a_i, b_i)$$



Abdolrashidi et al. Pareto-Optimal Quantized ResNet Is Mostly 4-bit
Efficient Deep Learning for Computer Vision @ CVPR 2021
https://arxiv.org/abs/2105.03536

Across Hardware or Architecture or Algorithm

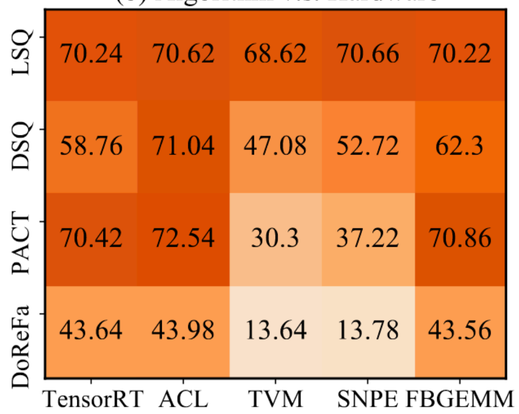
Table 2: Comparison of (1) the different hardware we selected and (2) the different QAT algorithms. *Infer. Lib.* is the inference library; *FBN* means whether fold BN. ❋ means undeployable originally, but can be deployable when certain requirements are satisfied.

Infer. Lib.	Provider	HW Type	Hardware	*s* Form.	Granularity	Symmetry	Graph	FBN
TensorRT [22]	NVIDIA	GPU	Tesla T4/P4	FP32	Per-channel	Symmetric	2	✓
ACL [23]	HUAWEI	ASIC	Ascend310	FP32	Per-channel	Asymmetric	1	✓
TVM [25]	OctoML	CPU	ARM	POT	Per-tensor	Symmetric	3	✓
SNPE [24]	Qualcomm	DSP	Snapdragon	FP32	Per-tensor	Asymmetric	3	✓
FBGEMM [26]	Facebook	CPU	X86	FP32	Per-channel	Asymmetric	3	✓
Algorithms	**Deployable**	**Uniformity**	**Quant. Type**	***s* Form.**	**Granularity**	**Symmetry**	**Graph**	**FBN**
LSQ [15]	❋	Uniform	learning-based	FP32	Per-tensor	Symmetric	1	✗
APoT [27]	✗	Non-uniform	learning-based	FP32	Per-tensor	Symmetric	1	✗
QIL [18]	✗	Uniform	learning-based	FP32	Per-tensor	Symmetric	1	✗
DSQ [28]	❋	Uniform	rule-based	FP32	Per-tensor	Symmetric	1	✗
LQ-Net [29]	✗	Non-uniform	rule-based	FP32	Per-tensor	Symmetric	1	✗
PACT [30]	❋	Uniform	learning+rule	FP32	Per-tensor	Symmetric	1	✗
DoReFa [31]	❋	Uniform	rule-based	FP32	Per-tensor	Symmetric	1	✗

Method	Architecture	FP Accuracy	Weight Bit-width	Activation Bit-width	Quantized Accuracy
Learned Step Size Quantization	ResNet-50	77.0%	4	4	76.3%
PACT: Parameterized Clipping Activation for Quantized Neural Networks	ResNet-50	77.0%	4	4	76.3%
DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients	ResNet-50	77.0%	4	4	76.2%
Differentiable Soft Quantization: Bridging Full-Precision and Low-Bit Neural Networks	ResNet-50	77.0%	4	4	74.8%
Learned Step Size Quantization	RegNetX-600MF	73.7%	4	4	72.5%

(a) Algorithm v.s. Architectures

(b) Algorithm v.s. Hardware



http://mqbench.tech/

Li et al. MQBench: Towards Reproducible and Deployable Model Quantization Benchmark
NeurIPS Dataset and benchmarks track 2021
https://openreview.net/pdf?id=TUplOmF8DsM

Quantization in action: Duolingo

birdbrain v2: productionization

1. Model Training
2. Model decomposition

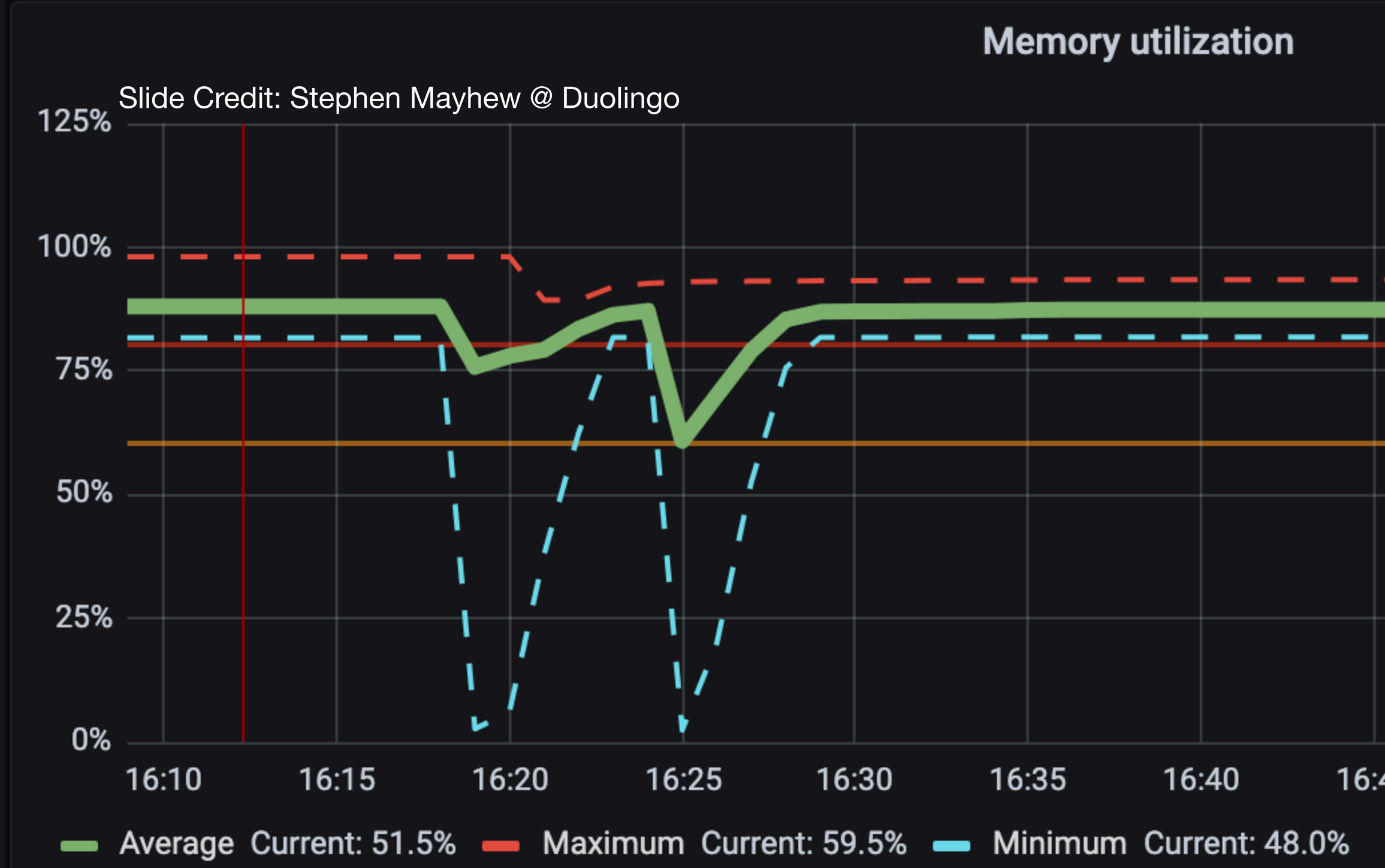
Different model for every course (100+ models)

Offline training job (in Python)

Trace to Torchscript

Quantize to 8-bit ints

Slide Credit: Stephen Mayhew @ Duolingo

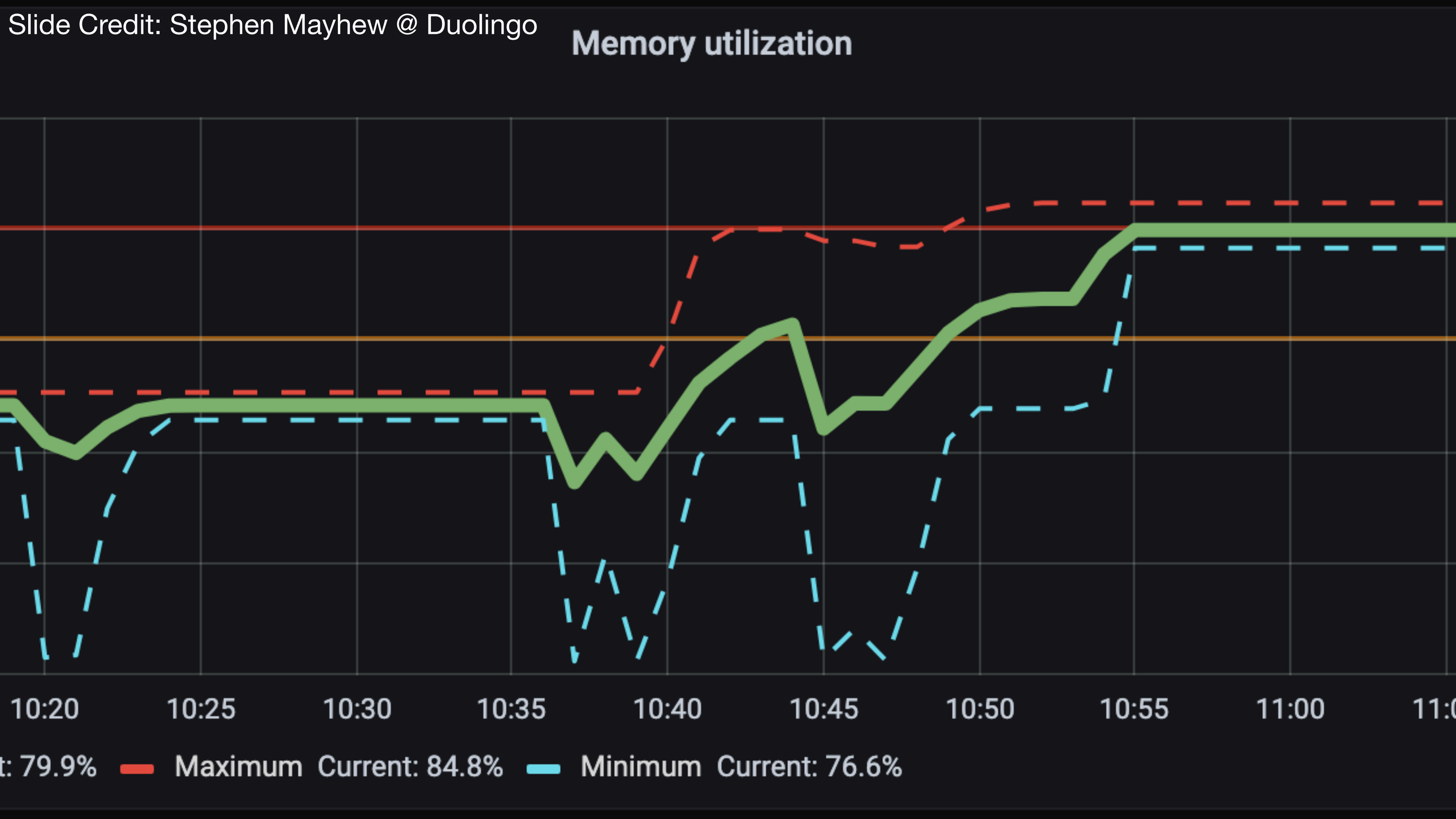

Memory utilization
Slide Credit: Stephen Mayhew @ Duolingo
125%
100%
75%
50%
25%
0%
16:10
16:15
16:20
16:25
16:30
16:35
16:40
Average Current: 51.5%
Maximum Current: 59.5%
Minimum Current: 48.0%

birdbrain v2: model A/B testing

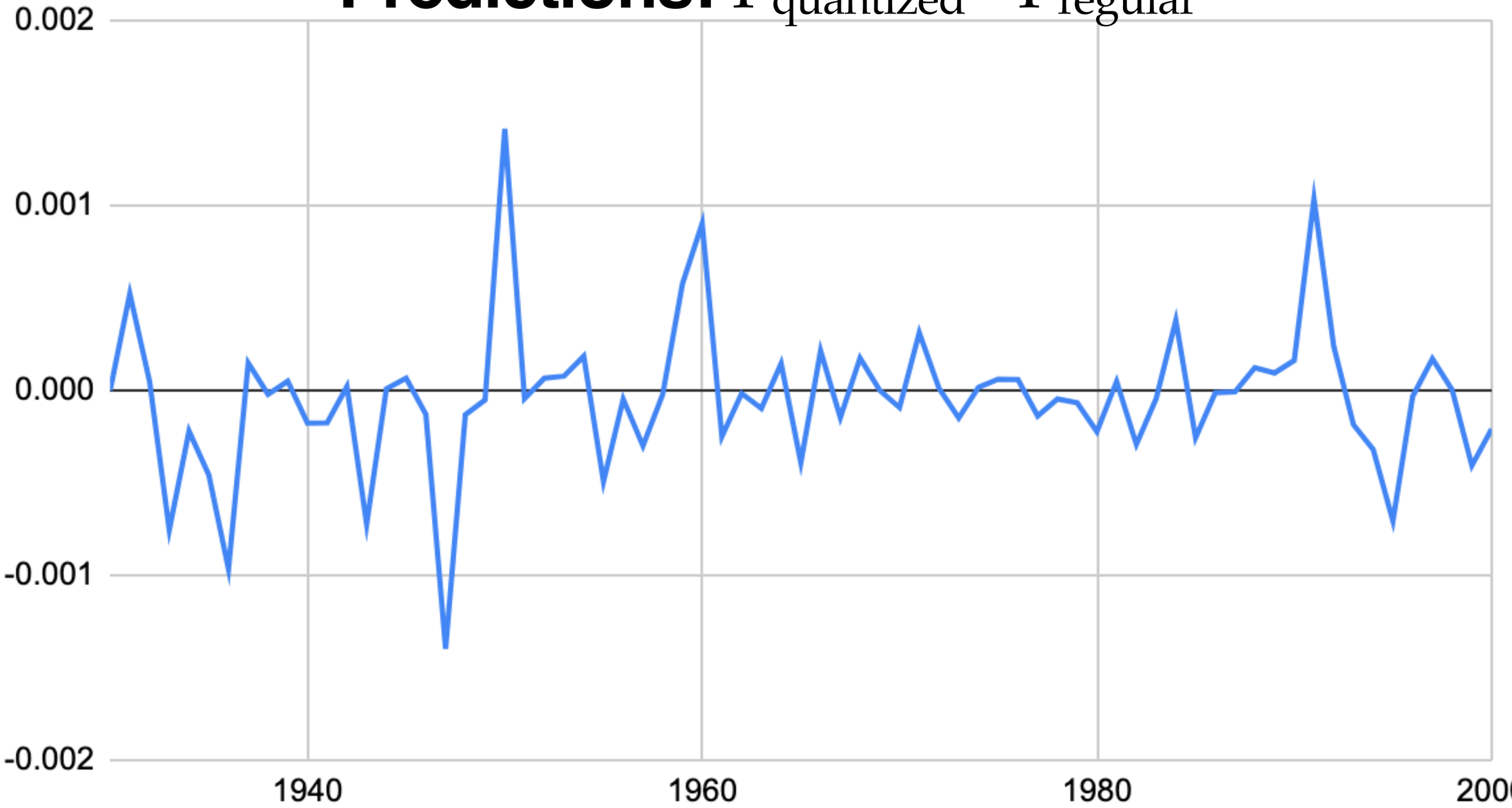
- We need the capability to A/B test models
- Reasons
  - When we refresh models (increasing OOV rate)
  - When we create new models
- Technical solution: just load twice the models

Slide Credit: Stephen Mayhew @ Duolingo

Slide Credit: Stephen Mayhew @ Duolingo



Predictions: $P_{quantized} - P_{regular}$



Slide Credit: Stephen Mayhew @ Duolingo

What about GPT-4?