Efficient Embedded Intelligence: Exploring the Width–Precision Trade-Off in Binary-Quantized Vision Transformers

Author
Ivar van Loon[1]

Supervisor
Braden Refalo[1]

Responsible Professor
Qing Wang[1]

[1]EEMCS, Delft University of Technology

Introduction

- Vision Transformers (ViTs) achieve state-of-the-art performance, but are **computationally expensive**, hurting embedded deployment capability.
- **Binary Quantization** reduces memory (**~32×**) and compute (**~64×**) significantly [1], replacing expensive **MatMul** operations with **XNOR + popcount** (Figure 1).
- Can we spend the **memory/compute budget** freed by binary quantization on **model width** to recover the lost accuracy?

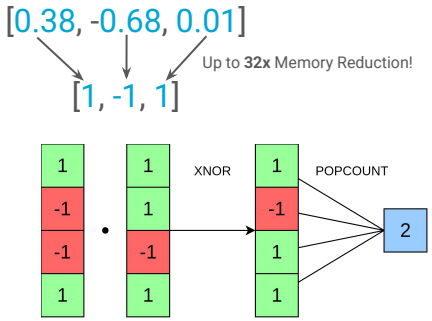


Figure 1: The dot product of 2 binary vectors can be implemented using the Bitwise Operations XNOR and POPCOUNT XNOR and POPCOUNT. Because a 64-bit CPU packs 64 of these into one instruction, this yields up to 64× savings over FP multiply-add.

Background

- Current SOTA Binary ViT model ⇒ **BHViT** [1].
- BHViT introduces a set of architectural and optimization changes to reduce the accuracy gap caused by binarization.
  - Does increasing width close this gap?

References

[1] T. Gao et al., "BHViT: Binarized Hybrid Vision Transformer," 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 2025, pp. 3563-3572, doi: 10.1109/CVPR52734.2025.00337.

Research Question

Can a **wider, low-bit** ViT match a **narrower full-precision** ViT under equivalent **compute** and **memory** constraints?

In other words, can **increasing model width** compensate for the **accuracy** lost through **binarization**, while preserving the **efficiency benefits** of binary transformers?

Method

1. Establish **FP Baseline** with the **BHViT-tiny** architecture.
   - This defines the Memory Budget M_{FP} and Compute Budget C_{FP}.
2. Compare **Binary BHViT** to **FP Baseline** to establish **accuracy gap** caused by quantization.
3. Scale **width** $w \in \{1\times, 2\times, 3\times\}$, a single multiplier on the **hidden** and **MLP** dimensions of the **Binary model** within the budgets M_{FP} and C_{FP} to try and close the accuracy gap.

To ensure a fair comparison, each model is trained on the **Oxford-IIIT Pet Dataset (37 categories)** for **2000 epochs** over 2 phases, where *Phase 1* trains an FP model **from scratch** and *Phase 2* **warm-starts** each binary model from its Phase 1 counterpart.

Savings remain **theoretical**: to our knowledge, no public framework provides bitwise kernels for ViTs yet.

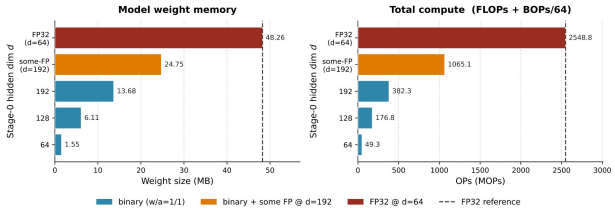
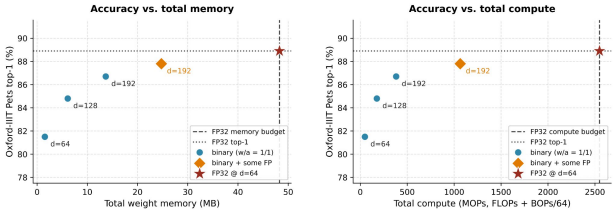


Figure 2: Theoretical weight memory (left) and compute (right) required for each model variant during inference.

Results



Model	w (d)	Mem. (MB)	OPs (M)	Top-1 (%)
FP Baseline	1 (64)	48.26	2548.8	88.9
Binary	1 (64)	1.55	49.3	81.8 (−7.1)
Binary	2 (128)	6.11	176.8	84.8 (−4.1)
Binary	3 (192)	13.68	382.3	86.7 (−2.2)
Binary†	3 (192)	24.75	1065.1	**87.8** (−1.1)

Figure 3: Classification accuracy plotted against weight memory and compute (top), showing an increasing but diminishing curve. Precise values given in the table (bottom), where † denotes that the downsampling layers were kept in full precision.

- **Tripling** width recovers **4.9** of **7.1** points of **Top-1** accuracy.
  - Retaining a **3.5×** reduction in weight memory and **6.7×** reduction in compute during inference (Figure 2, 3).
  - Keeping **downsampling layers** in **FP** recovers a further **1.1** points of accuracy, suggesting that the gap is partly a **precision bottleneck** that width alone may not fix.

Conclusion & Future Work

- Reinvesting quantization savings into width is an **effective** strategy, bringing binary ViTs within **~1–2 points** of full precision at a **fraction of the cost**.
- Future work could explore **larger widths** on better hardware, validate on **large-scale datasets**, and build **optimized 1-bit kernels** to realize the efficiency gains in practice.

Code is available at: https://github.com/ivarvl/bhvit-width-scaling
Contact: I.S.vanLoon@student.tudelft.nl