Efficient Embedded Intelligence

Exploring the Width-Precision Trade-Off in Binary-Quantized Vision Transformers

Ivar van Loon[1]

Supervisor(s): Braden Refalo[1]**, Qing Wang**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Ivar van Loon
Final project course: CSE3000 Research Project
Thesis committee: Braden Refalo, Qing Wang, Julia Olkhovskaia

Abstract

Vision Transformers perform strongly across computer vision tasks but often require too much compute and memory for embedded deployment. Binary quantization cuts these costs by constraining weights and activations to a single bit, at the expense of accuracy. We investigate whether the budget freed by binarization can be reinvested into additional model width to recover that lost accuracy. Using the BHViT-Tiny architecture on the Oxford-IIIT Pet dataset, we first isolate the accuracy gap caused by quantization alone by comparing a full-precision reference against its binarized counterpart at identical width, and then scale width within the freed budget to measure how much of this gap can be recovered by width. We find that binarization at the base width costs 7.1 points of Top-1 accuracy, and that tripling the width recovers 4.9 of these points while remaining at a theoretical $3.5\times$ and $6.7\times$ reduction in memory and compute relative to the full-precision reference. The wider binary model thus approaches full-precision accuracy at a fraction of its cost. Additionally, keeping the downsampling layers in full precision recovers a further 1.1 points at a cost still well within budget, narrowing the gap to 1.1 points and indicating that part of the residual loss stems from a precision bottleneck rather than from a global lack of capacity. Our results establish width scaling as an effective strategy for reducing the binarization accuracy gap, offering a promising path toward the resource-constrained deployment of Vision Transformers. The code to reproduce all experiments is available on GitHub[1].

1 Introduction

Introduced in *Attention Is All You Need* [17], transformer models rely on attention mechanisms rather than the recurrence of earlier sequence models, enabling efficient parallel computation and strong modelling of long-range dependencies. This led to major advances in natural language processing, as well as computer vision through Vision Transformers (ViTs) [4]. ViTs perform better and scale more effectively than traditional Convolutional Neural Networks (CNNs), but this comes at a cost: the quadratic complexity of self-attention with respect to input size results in high computational and memory demands [15]. This is especially limiting in embedded and edge settings, such as drones, mobile devices, or wearable devices, where models must operate under tight power, memory, and latency budgets and where offloading to the cloud is often made impossible by real-time constraints, privacy requirements, or unreliable connectivity. Much recent research therefore targets transformer efficiency through techniques such as quantization, which represents weights and/or activations at lower numerical precision. Extreme quantization can substantially reduce memory usage and inference cost, but often at the expense of accuracy [6].

[1] https://github.com/ivarvl/bhvit-width-scaling

Recent work on extreme quantization has explored *binary* and *ternary* quantization, where weights and/or activations are constrained to low-bit representations. Architectures such as *BitNet* for LLMs [10] and *BHViT* for vision [5] demonstrate that low-bit transformers can achieve competitive performance while significantly reducing computational requirements. These results suggest that reduced numerical precision may be compensated by architectural or training modifications.

One important but relatively under-explored question is whether increasing model width can offset the loss in representational capacity caused by aggressive quantization [18; 14]. Motivated by this question, this paper investigates the trade-off between model width and numerical precision in Vision Transformers under fixed compute and memory constraints. The central research question of this work is:

> *Can a wider, low-bit Vision Transformer match or exceed the task performance of a narrower, full-precision Vision Transformer, when both models are constrained to equivalent compute and memory budgets?*

To answer this question, we first establish a full-precision (FP) baseline and measure the impact of low-bit quantization on classification accuracy. Next, we analyse how increasing the width of low-bit Vision Transformers affects performance while not exceeding the computational cost of the FP model. Then, we conduct an ablation study to measure the accuracy gain of each individual approach of increasing model width. Finally, we evaluate the practical deployment potential of the resulting models on a Raspberry Pi 5, a representative low-power embedded platform, by measuring inference latency and memory usage. Since, to our knowledge, no publicly available inference framework yet offers the optimized bitwise kernels required for efficient execution of binary Vision Transformers, these measurements do not reflect the full theoretical speedup of binarization. They instead represent a lower bound on the efficiency these models could achieve once such kernels exist.

2 Background

In this section, we review the standard Vision Transformer (Section 2.1), introduce binary quantization and its effect on cost and accuracy (Section 2.2), and describe the BHViT Architecture [5] (Section 2.3), the binary model that serves as the base for our study.

2.1 The Vision Transformer

The standard Vision Transformer (ViT), as described by Dosovitskiy et al. [4], works by first splitting up the input image into N patches, each of size $P \times P$, where P is the *patch size*. Each patch contains $P^2 \cdot C$ values ($C = 3$ for an RGB image), which are flattened to a 1-D vector. The flattened patch vectors are then multiplied with a learned weight matrix, called the *Patch Embedding Matrix*, denoted $\mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}$, where D is referred to as the *Hidden Dimension* or *Embedding Dimension*. A learnable vector $\mathbf{z}_{class} \in \mathbb{R}^D$, known as the *class token* (or *class embedding*), is prepended to the resulting patch embeddings, forming a

sequence of length $N+1$. A learned *Positional Embedding Matrix*, denoted $\mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D}$, is then added to this full sequence. This can be summarized as:

$$\mathbf{Z}_0 = [\mathbf{z}_{class} \mid \mathbf{X}_p\mathbf{E}] + \mathbf{E}_{pos} \tag{1}$$

where $\mathbf{X}_p \in \mathbb{R}^{N\times(P^2\cdot C)}$ is the matrix of all flattened patches stacked, and $\mathbf{Z}_0 \in \mathbb{R}^{(N+1)\times D}$ is the resulting sequence of embedded patches including the class token. This sequence is then fed to the *Transformer Encoder* [17], after which we can determine the classification result by taking the final hidden state corresponding to the class token and passing it through a classification head, typically a Multi-Layer Perceptron (MLP).

The transformer encoder is a stack of identical blocks, each pairing the self-attention mechanism with an MLP, with residual connections and layer normalization applied around both. The key component is the *self-attention* mechanism, which allows every image patch to attend to every other patch in the image. Given the embedded sequence $\mathbf{Z}$ produced above, three matrices are computed: the query matrix $\mathbf{Q}$, key matrix $\mathbf{K}$, and value matrix $\mathbf{V}$:

$$\mathbf{Q} = \mathbf{Z}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{Z}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{Z}\mathbf{W}_V \tag{2}$$

where $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are learned projection matrices. The *Attention Matrix* $\mathbf{A}$, which holds the weight each patch assigns to every other patch, and the attention output are then computed as:

$$\mathbf{A} = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right), \qquad \mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathbf{A}\mathbf{V} \tag{3}$$

where d_k is the dimension of the key vectors.

In practice this is done with *Multi-Head Attention*: the computation above is carried out in parallel by several *heads*, each with its own projection matrices, and their outputs are concatenated. This lets the model learn different types of relationships between patches simultaneously.

The second sub-block is a position-wise MLP, also known as the *feed-forward* network, applied identically and independently to each token in the sequence:

$$\mathrm{MLP}(\mathbf{Z}) = \sigma(\mathbf{Z}\mathbf{W}_1 + \mathbf{b}_1)\,\mathbf{W}_2 + \mathbf{b}_2 \tag{4}$$

where $\mathbf{W}_1 \in \mathbb{R}^{D\times D_{\mathrm{mlp}}}$ and $\mathbf{W}_2 \in \mathbb{R}^{D_{\mathrm{mlp}}\times D}$ are learned weight matrices, $\mathbf{b}_1$ and $\mathbf{b}_2$ the corresponding biases, and σ the GELU nonlinearity. The block first projects each token from the hidden dimension D up to a larger *MLP intermediate dimension* D_{mlp}, applies the nonlinearity, and projects back down to D, so that the input and output dimensions match and the residual connection can be applied. In the standard ViT this intermediate dimension is set to $D_{\mathrm{mlp}} = 4D$.

2.2 Binary Quantization

Quantization improves the computational efficiency of deep learning models by reducing the numerical precision of weights and/or activations. Standard transformers use 32-bit floating-point arithmetic, which is precise but memory- and compute-intensive. Lowering this precision reduces memory consumption and speeds up inference, making models deployable on resource-constrained devices like mobile and embedded hardware. These gains typically come at the cost of accuracy, however, since reduced precision introduces quantization error that worsens at lower bit-widths [6].

With *Binary Quantization*, the weights and/or activations can only be assigned two values, often $\{-1, 1\}$ or $\{0, 1\}$, depending on the type of operation. We can obtain the binarized representation by applying the following function to the FP representation:

$$\mathbf{W}_{\mathrm{B}} = \mathrm{sign}(\mathbf{W}_{\mathrm{FP}}) \tag{5}$$

where

$$\mathrm{sign}(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases} \tag{6}$$

However, the $\mathrm{sign}(\cdot)$ function creates a problem for training: it is a step function, and step functions are non-differentiable at their discontinuities. Its derivative is zero everywhere else and undefined at $x = 0$, so it provides no usable gradient. Since backpropagation relies on continuous, non-zero derivatives to propagate gradients via the chain rule, the binarized network cannot be trained directly.

Binary networks address this through *Quantization-Aware Training (QAT)*, which separates the *forward* and *backward* passes as follows:

- **Forward pass** The network operates as intended, weights and activations are binarized using $\mathrm{sign}(\cdot)$.
- **Backward pass** Rather than updating the binary weights directly, a parallel set of full-precision *latent weights* is maintained. These are what the optimizer actually updates.

Gradients must still pass *through* the sign operator during backpropagation to reach the latent weights. To enable this, a *Straight-Through Estimator* (STE) is used [1]. The STE simply pretends that the sign function was never applied during the backward pass, treating it as if it were the identity function instead: $f(x) = x$.

By restricting weights to binary values, expensive floating-point matrix multiplications and vector dot products can be replaced by significantly cheaper bitwise operations, XNOR and POPCOUNT, as illustrated in Figure 1 [13]. We can express the computational cost in a unified metric, first defined in Bi-Real Net [9] and later adopted by Liu et al. [8]:

$$\mathrm{OPs} = \mathrm{FLOPs} + \frac{\mathrm{BOPs}}{64}, \tag{7}$$

where FLOPs denotes floating-point operations and BOPs denotes bitwise operations. The factor of 64 arises because a 64-bit CPU executes XNOR and POPCOUNT over 64 bits in a single instruction each, so 64 BOPs are performed at approximately the cost of one FLOP [9]. Furthermore, binary models require far less weight memory compared to FP32 representations, theoretically reducing memory usage by a factor of 32.

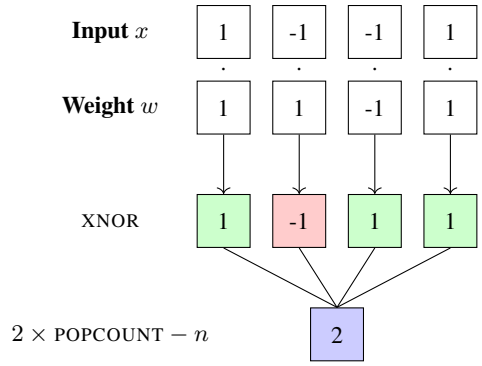

Figure 1: Example of binary vector dot product computation using XNOR and POPCOUNT. Matching bits produce a value of 1 after the XNOR operation, while mismatched bits produce a value of -1. The POPCOUNT operation counts the number of matching bits. Since each match contributes $+1$ and each mismatch contributes -1, the final dot product is recovered using $2 \times$ POPCOUNT $- n$, where n is the vector length. In this example, POPCOUNT $= 3$ and $n = 4$, resulting in a dot product of $2 \times 3 - 4 = 2$.

Table 1: Accuracy drop caused by binarization of weights and activations with ReActNet and BHViT, trained on ImageNet-1K. Full-Precision refers to 32-bit Floating Point precision. (Table adapted from BHViT [5]).

Architecture	Quantization	Top-1	Δ
ResNet-34	Full-Precision	73.3	
	Binary (ReActNet [8])	69.3	$\mathbf{-4.0}$
DeiT-Small	Full-Precision	79.9	
BHViT-Small	Full-Precision	79.3	
BHViT-Small	Binary (BHViT [5])	68.4	$\mathbf{-10.9}$
DeiT-Small	Binary (ReActNet [8])	49.5	-30.4

These efficiency gains, however, come at the cost of reduced representational capacity, and therefore lower predictive performance. Because binary weights and activations can only encode two discrete values, the model loses much of the expressive power available in full-precision networks. Table 1 highlights this trade-off on the ImageNet-1K benchmark, in which *both* weights and activations are quantized to 1 bit. While ReActNet applied to ResNet-34 achieves competitive performance with only a 4.0 percentage point drop in Top-1 accuracy compared to its full-precision counterpart, the accuracy degradation becomes substantially larger for transformer architectures. The current state-of-the-art binarization method for ViTs, BHViT, demonstrates a 10.9 percentage point reduction in Top-1 accuracy relative to its full-precision counterpart.

2.3 The BHViT Architecture

BHViT [5] already addresses some of the accuracy loss of binarized ViTs through a set of targeted architectural and optimization changes. Rather than binarizing a standard ViT directly, it redesigns the network so that it is easier to train and retains more of its accuracy, while keeping nearly all weights and activations binary, technically making it a mixed-precision model. A brief background on each of these architectural optimizations is given below. For a complete description we refer the reader to the original paper, *BHViT: Binarized Hybrid Vision Transformer*, by Gao et al. [5]

Feature Pyramid BHViT utilizes a four-stage *Feature Pyramid* structure. An initial patch-embedding stem, a binary convolution with a 4×4 kernel and a stride of 4, projects the three-channel input image to a 64-channel feature map at one-quarter spatial resolution ($H/4 \times W/4$). The four stages then operate at channel widths of 64, 128, 256, and 512, with the spatial resolution falling from $H/4$ to $H/32$: after each stage a binary convolutional downsampling layer with a 2×2 kernel and a stride of 2 halves the spatial resolution of the feature map and doubles the channel dimension. This produces a hierarchy similar to convolutional backbones such as ResNet. Since binary self-attention degrades when the token count is large and is expensive at high spatial resolutions, the first two stages use a convolution-based token mixer, switching to attention only in the final two stages, once the number of tokens has been sufficiently reduced by the downsampling layers.

Token Mixing The token mixer is the component that lets different image patches exchange information, and BHViT uses two variants. In the first two stages it applies the *Multi-Scale Grouped Dilated Convolution* (MSGDC) module: several grouped convolutions in parallel, each with a different *dilation rate* so that each captures patterns at a different scale, far more cheaply than attention.

The final two stages use the *Multi-Scale Multi-Head Attention* (MSMHA) module, a windowed form of self-attention. Rather than letting every token attend to every other, the feature map is divided into fixed-size windows and attention is computed within each. To retain global context, a coarse version of the whole feature map, obtained by average-pooling it into a small grid, is appended to every window, so each window attends both to its own tokens and to a compressed summary of the entire image. This reduces the number of tokens in any single computation while preserving much of the accuracy, resulting in a more efficient model.

Quantization Decomposition Binarizing the *attention matrix* $\mathbf{A}$ (Equation 3) loses too much expressiveness: each entry collapses to 0 or 1, so a token can only fully attend to another or ignore it entirely, with nothing in between. To solve this, BHViT introduces *Quantization Decomposition* (QD). Since $\mathrm{softmax}$ already puts every attention value in the range $[0, 1]$, QD scales the matrix by a scaling factor $s = 3$, and rounds the result, mapping each entry to one of a handful of discrete levels. That level is then represented as a stack of binary matrices. Adding the stack back together recovers the original level, letting a token attend to another weakly, moderately, or strongly. Crucially, every matrix in the stack is still binary, keeping the computations as efficient as possible.

Dense Residual Connections Because the *sign* function has no usable gradient, binary networks rely on a *straight-through estimator*, which substitutes a usable gradient during backpropagation: it lets the gradient pass through unchanged when a weight or activation lies within a bounded interval (e.g., $[-1, 1]$), and sets it to zero once the value drifts outside

it, in order to prevent unnecessary weight updates. The gradient reaching a deep weight is a product of many such truncated terms, one for each binary layer it must pass through, and this product quickly collapses toward zero, so the early layers receive almost no learning signal. To mitigate this, BHViT adds shortcuts inside each attention block, adding the query, key, and value activations back to the attention output before they are binarized. Now when the binarized layers pass no gradient, a usable signal still reaches the weights, keeping the binary layers trainable.

Binary MLP and Shift Operations Each block contains a binary *Multi-Layer Perceptron* (MLP) that mixes information across channels. Since binarizing it hurts its capacity, BHViT augments it with inexpensive *shift operations* that move a feature map by one position horizontally or vertically, or borrow a fraction of the channels from neighbouring tokens. Requiring no multiplications, these reintroduce some of the spatial and cross-channel information lost to binarization at negligible cost.

Training Strategy Two further measures are applied during training. BHViT is trained with *knowledge distillation*, using a full-precision DeiT teacher whose soft predictions guide the binary student more informatively than hard class labels [7; 16]. BHViT also adds a regularization loss in the final 10% of epochs to counteract *weight oscillation*: the full-precision latent weights maintained during training tend to settle near zero, where they repeatedly flip sign without progress. BHViT defines this loss as

$$\mathcal{L}_{re} = \frac{1}{n}\sum_{i=1}^{n} \big|\, |w_i| - 1 \,\big|, \tag{8}$$

which, unlike ordinary weight decay, is minimized when each latent weight reaches $+1$ or -1 and maximized at zero. It therefore pushes the latent weights out of the unstable near-zero region during the final epochs, stabilizing training so that more weights continue to update meaningfully.

3 Method

This section formalizes the width-precision trade-off. We frame it as a fixed-budget design choice, reinvesting the memory and compute freed by binarization into additional model width (Section 3.1). We then compare a full-precision reference against its binarized counterpart at equal width to isolate the accuracy gap from quantization (Section 3.2), and scale width within the fixed budget to close it (Section 3.3).

3.1 Reinvesting Quantization Savings into Width

Storing model weights and activations using a single bit shrinks its memory footprint by up to a factor of 32, and enables the replacement of floating-point matrix multiplications with bitwise XNOR and POPCOUNT operations [13], reducing compute under the unified cost metric of Equation 7. As established in Section 2.2, however, these savings come at the price of representational capacity, and accuracy drops accordingly.

Rather than deploying a narrow binary network that is both cheaper and less accurate than its full-precision counterpart, we can reinvest some of the recovered memory and compute into a *wider* binary model [14; 18]. Concretely, we identify width with the dimensions of the transformer blocks introduced in Section 2.1: increasing w scales both the hidden dimension D and the intermediate (MLP, also called *feed-forward*) dimension D_{mlp}, so a wider model devotes more channels to representing each token and a larger projection within each feed-forward sublayer. Formally, let $M(\cdot)$ and $C(\cdot)$ denote the weight memory and compute of a model. Given a full-precision reference with budget M_{FP} and C_{FP}, we constrain every binary candidate of width w to:

$$\begin{aligned} M_{\text{bin}}(w) &\le M_{\text{FP}}, \\ C_{\text{bin}}(w) &\le C_{\text{FP}}, \end{aligned} \tag{9}$$

and ask whether, within this constraint, a wider low-bit model can match or exceed the accuracy of the narrow full-precision reference.

3.2 Isolating the Quantization Gap

To attribute any difference in accuracy to numerical precision alone, we hold the architecture fixed across the comparison. We train a *full-precision* instance of the BHViT-Tiny architecture (Section 2.3) at the base width, which serves two purposes: it provides the accuracy reference, and its memory and compute define the budget M_{FP} and C_{FP} of Equation 9. The components of BHViT that exist solely to support binarization carry no meaning at full precision; therefore, they are disabled in this FP baseline (Section 4.2).

We then construct the binarized counterpart by applying the BHViT binarization scheme (Sections 2.2 and 2.3) to the *same* architecture at the *same* width, constraining the weights and activations throughout the model to a single bit while leaving the architecture untouched. Since the two models differ only in numerical precision, the difference in their accuracy measures the quantization gap that width scaling must subsequently recover. This base-width binary model is the direct quantized counterpart of the reference and the starting point of the width-scaling study.

To improve the accuracy of the binarized models, we do not train each binary model from scratch. Instead, we initialize it from the converged full-precision counterpart of the same width, an established technique for improving the accuracy of binary networks, which are difficult to optimize directly [11]. Because this protocol requires a full-precision model at each width, we train a full-precision instance for every width in the sweep. These per-width FP models exist solely to initialize their binary counterparts, and only the base-width full-precision model serves as the accuracy reference and defines the budget of Equation 9.

To keep the comparison fair in terms of training budget, the full-precision baseline uses a warm-start as well. Therefore, the baseline and every binary model receive the same amount of additional training, so that any accuracy difference reflects numerical precision and capacity rather than an asymmetry in optimization effort. The precise schedule is given in Section 4.2.

Table 2: Per-stage dimensions of BHViT, scaled by multiplier w.

	Stage 0	Stage 1	Stage 2	Stage 3
Hidden	$64w$	$128w$	$256w$	$512w$
Intermediate	$512w$	$1024w$	$1024w$	$2048w$
Expansion Ratio	$8\times$	$8\times$	$4\times$	$4\times$

3.3 Recovering Accuracy through Width Scaling

To recover the lost accuracy, we widen the binary network through a single scalar multiplier w applied uniformly to every per-stage channel dimension, scaling both the hidden (embedding) and MLP intermediate dimensions: because both dimensions scale by the same factor, the per-stage MLP expansion ratios (D_{mlp}/D_{hidden}, Table 2) stay the same, and every other architectural and optimization hyperparameter is left unchanged. Width is therefore the only degree of freedom in the sweep, ensuring that any change in accuracy is attributable to capacity rather than to a confounding change in depth, expansion ratio, attention structure, or optimization.

While the inference budget of Equation 9 allows for significant expansion, our exploration is practically limited by the VRAM requirements of training rather than by inference cost. Because quantization-aware training maintains full-precision latent weights (Section 2.2), training a width-scaled binary model is as memory-intensive as training a full-precision model of the same width, even though its eventual inference cost stays well within the deployment budget. This cost is further increased by the warm-start protocol of Section 3.2, which requires training an additional full-precision model at each width to supply the initialization. Training hardware, rather than the inference budget, is therefore the practical limit on the largest width we explore.

The multiplier w covers two distinct sources of capacity. The embedding dimension widens every layer, including the attention projections and the token representation itself, whereas the feed-forward intermediate dimension widens only the MLP blocks, and the two load the memory and compute budgets differently. To attribute the recovered accuracy to its source, we perform an ablation study that scales the hidden and intermediate dimensions independently, measuring the contribution of each.

4 Experimental Setup

This section provides the concrete dataset (Section 4.1), architecture configuration and training procedure (Section 4.2), and width-scaled variants (Section 4.3) used to instantiate the method of Section 3.

4.1 Dataset and Augmentation

All experiments are conducted on the *Oxford-IIIT Pet dataset* [12], a fine-grained image classification benchmark containing 7,349 images spread across 37 pet breeds (12 cat and 25 dog breeds). The images exhibit large variations in scale, pose, and lighting, which makes classification a non-trivial task despite the relatively small number of classes. We adopt the standard split defined by the dataset authors, using the 3,680 `trainval` images for training and the 3,669 `test` images for evaluation.

We select this dataset because its small size makes it feasible to train multiple Vision Transformer variants from the same fixed training budget, allowing a controlled comparison across model widths without the cloud compute cost of training many models on a large-scale dataset such as ImageNet-1K.

All images are processed at a resolution of 224×224. During training we follow the data-augmentation procedure commonly used for Vision Transformers and adopted by BHViT: random resized cropping, random horizontal flipping, RandAugment [2], colour jitter, random erasing, Mixup and CutMix.

4.2 Architecture and Training Configuration

Both the full-precision reference and its narrow binarized counterpart use the BHViT-Tiny configuration with a base stage-0 width of $d = 64$, with the same DeiT-Tiny teacher model used for distillation. The full architecture and training configuration are summarized in Appendix A. Besides model width and numerical precision, the same settings are used for all full-precision, binarized, and width-scaled variants. This ensures that any change in accuracy can be attributed to quantization or capacity rather than to a difference in architecture or hyperparameters.

All models are trained in two phases, following the warm-start protocol of Section 3.2. In *Phase 1*, a full-precision model is trained from scratch for $1{,}000$ epochs. In *Phase 2*, the model is trained for a further $1{,}000$ epochs: the full-precision reference resumes from its own converged checkpoint, while each binary model is initialized from the converged full-precision model of the same width and then trained with quantization-aware training. Because the same two-phase budget is applied to both the reference and binary models, any accuracy difference still reflects numerical precision and capacity rather than a difference in training effort. Each configuration is trained three times with three different seeds, and we report the mean Top-1 accuracy across the three runs.

When training the full-precision models, the components specific to binary networks carry no meaning and are disabled, namely: the RPReLU activation recalibration [8], the binarization-aware regularization loss, and the optional floating-point attention path.

For the binarized models, each component is binarized as described in the BHViT paper [5], and the RPReLU activations and binarization-aware regularization loss are enabled, while the optional floating-point attention path remains disabled, and is instead considered separately. Following standard practice for binary networks, a small number of layers are always kept in full precision, as binarizing them is known to cause disproportionate accuracy loss: the input patch-embedding convolution, the final classification head, and all normalization parameters. Every other layer in the network is binary, making BHViT a mixed-precision model.

4.3 Width-Scaled Variants

We instantiate the width multiplier of Table 2 at $w \in \{1\times, 2\times, 3\times\}$, corresponding to stage-0 hidden dimensions $d \in \{64, 128, 192\}$, and additionally evaluate a variant at

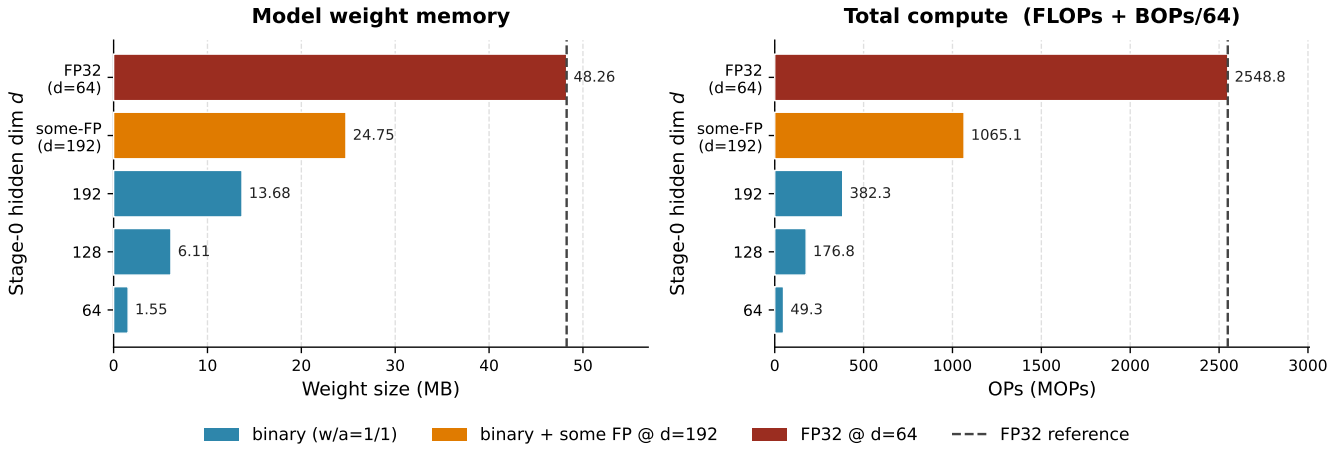

Figure 2: Theoretical weight memory (left) and total compute (right) of the binary BHViT across the stage-0 hidden-dimension sweep $d \in \{64, 128, 192\}$, compared against the FP32 baseline at $d = 64$ (red, dashed reference line). Binary models use 1-bit weights and activations (w/a = 1/1). Even the largest binary configuration ($d = 192$) requires only 13.7 MB and 382 MOPs, a 3.5× and 6.7× reduction over the FP32 baseline (48.3 MB, 2549 MOPs), respectively.

$d = 192$ in which the downsampling layers between model stages retain full precision. The resulting weight memory and operation count for each candidate are reported in Figure 2. All four candidates satisfy the budget of Equation 9: the widest fully binary configuration ($w = 3$, $d = 192$) theoretically requires only 13.7 MB and 382 MOPs for inference, a 3.5× and 6.7× reduction over the full-precision reference, while the variant at $d = 192$ with full-precision downsampling layers requires 24.75 MB and 1065 MOPs, still within budget, though at a considerably higher cost than the fully binary counterpart. Each width follows the two-phase schedule of Section 4.2. We train all widths from $w = 1$ to $w = 3$ under the identical configuration described in Section 4.2, and select $w^* = 3$ as the largest width our training hardware supports: as noted in Section 3.3, training cost scales with width as if the model were full precision, and the protocol additionally requires training a full-precision model at every width.

5 Results

In this section, we first isolate the accuracy loss attributable to binarization alone by comparing a full-precision and binary model of identical architecture (Section 5.1), then investigate how width scaling can recover that loss within the same computational budget (Section 5.2), ablate the individual contributions of hidden and intermediate size to the observed accuracy gains (Section 5.3), and finally evaluate the practical inference efficiency of the resulting models on a Raspberry Pi 5 (Section 5.4).

5.1 The Quantization Gap

The full-precision narrow BHViT-Tiny baseline achieves 88.9% Top-1 accuracy on the Oxford-IIIT Pet test set. Applying the BHViT binarization scheme to the same architecture at the same base width ($w = 1$, $d = 64$), so that the two models differ only in numerical precision, reduces accuracy to 81.8%, a drop of 7.1 percentage points. This gap isolates the loss attributable to precision alone, since architecture, width, training schedule, and optimization are held identical across the pair (Section 3.2).

Table 3: Top-1 accuracy on the Oxford-IIIT Pet Dataset of the binary BHViT-Tiny across the width sweep, with per-variant weight memory and compute. All binary variants satisfy the budget $M_{\text{FP}} = 48.3\,\text{MB}$, $C_{\text{FP}} = 2.55\,\text{GOPs}$ of Equation 9. The full-precision reference (w=1, d=64) is shown for comparison. †: Downsampling layers retained in full precision.

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)

The magnitude of this drop defines the deficit that width scaling must subsequently recover. Notably, the binary model uses only 1.55 MB of weight memory and 49.3 MOPs per image, against the 48.3 MB and 2.55 GOPs of the reference (Figure 2), leaving a large fraction of the budget of Equation 9 unspent.

5.2 Recovering Accuracy through Width Scaling

We next train the binary model at each width $w \in \{1, 2, 3\}$, every variant warm-started from a converged full-precision model of matching width (Section 3.2) and trained under the identical configuration described in Section 4. Table 3 reports the Top-1 accuracy of each binary variant alongside its weight memory and compute, and Figure 3 plots accuracy against width. Increasing width raises the accuracy of the binary model from 81.8% at $w = 1$ to 86.7% at $w = 3$, a recovery of 4.9 percentage points over the base-width binary model. The trend is monotonic but diminishing. At $w = 3$ the binary model comes within 2.2 points of the full-precision reference of Section 5.1 but does not reach it.

Crucially, every width in the sweep remains well within

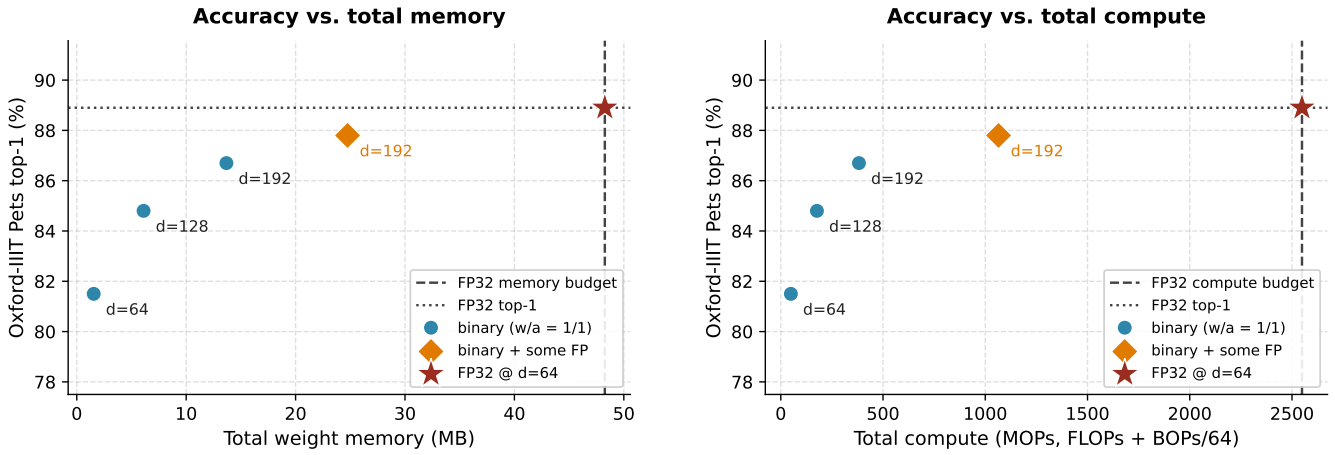

Figure 3: Top-1 accuracy on Oxford-IIIT Pet for fully binary BHViT models (w/a $= 1/1$, blue dots) as the stage-0 width d is swept, compared against an FP32 baseline at $d = 64$ (red star). **Left:** accuracy vs. total weight memory. **Right:** accuracy vs. total compute (FLOPs + BOPs$/64$). The dashed vertical line marks the FP32 budget, the dotted horizontal line marks FP32 top-1 accuracy. The orange diamond marks the *some FP* variant at w=3: identical to the fully binary model except that all downsampling layers are kept in full precision.

the budget of Equation 9: even the widest variant ($w = 3$) consumes only 13.7 MB and 382 MOPs, a 3.5× and 6.7× reduction relative to the full-precision reference (Figure 2). Under the equal-budget constraint that frames our research question, the width-for-precision exchange therefore does not fully succeed within the widths explored: tripling width recovers roughly two-thirds of the quantization gap (4.9 of 7.1 points) but leaves a 2.2-point gap compared to the full-precision reference.

The width sweep binarizes every layer uniformly. As an alternative use of the unspent budget, we evaluate the variant that keeps the downsampling layers between stages in full precision [5], applied to the widest (w=3) model (Table 3, denoted by †; Figure 3, coloured orange). This selective relaxation recovers a further 1.1 points over the fully binary w=3 model, reaching **87.8%** and narrowing the gap to the full-precision reference to 1.1 points. The cost is an increase to 24.75 MB and 1065 MOPs, both of which still remain within the budget of Equation 9 (1.9× and 2.4× reductions relative to the full-precision reference). Keeping these downsampling layers in full precision thus closes roughly half of the residual gap left by width scaling alone, at a memory and compute cost well below the full-precision model.

5.3 Ablation: Hidden vs Intermediate Size

Table 4 disentangles the contributions of the MLP intermediate size D_{mlp} and the attention hidden size D through an additive ablation on the fully binary BHViT-Tiny. Widening only the intermediate dimension to 3× raises Top-1 accuracy by 1.8 points (from 81.8% to 83.6%) at a modest cost of 2.84 MB and 51.9 M additional OPs. Subsequently widening the hidden dimension lifts accuracy by a further 3.1 points, but at a far steeper price: memory grows to 13.68 MB and OPs to 382.3 M.

5.4 Evaluation on the Raspberry Pi

We measure end-to-end inference latency and peak RAM on a Raspberry Pi 5 using ONNX Runtime (Appendix B) [3].

Table 4: Additive ablation isolating the contribution of the MLP intermediate size d_{i} versus the attention hidden size d_{h} for the fully binary ($w/a = 1/1$) BHViT-Tiny on Oxford-IIIT Pet. Starting from the base variant, we first widen only the intermediate sizes to measure its standalone effect, then additionally widen hidden sizes to measure the effect of both.

Variant	Mem. (MB)	OPs (M)	Top-1 (%)
Base (1×)	1.55	49.3	81.8
+ Intermediate (3×)	4.39	101.2	83.6 (+1.8)
+ Hidden (3×)	13.68	382.3	86.7 (+3.1)

To our knowledge, no optimized binary inference framework is publicly available on this platform; therefore, the binary models are deployed with INT8 quantization rather than true 1-bit operations, so the reported latencies represent the upper bound on what a 1-bit framework could achieve. At base width (d=64), INT8 lowers latency from 248 ms to 202 ms and RAM from 588 MB to 554 MB versus the FP baseline, whereas the wide model (w=3, d=192) runs at 651 ms and 835 MB, exceeding the FP baseline once its weights are quantized at 8 bits rather than 1 bit.

6 Discussion and Limitations

The central question of this work was whether a wider, low-bit Vision Transformer can match or exceed a narrower full-precision model under equal compute and memory budgets. Our results show that within the widths our training hardware allowed, width scaling recovers a substantial share of the accuracy lost to binarization, but it does not fully close the gap.

Tripling the width lifts the fully binary BHViT-Tiny from 81.8% to 86.7% on Oxford-IIIT Pet, recovering 4.9 of the 7.1 points that separate it from the 88.9% full-precision reference, while leaving a 2.2-point shortfall. What is gained instead is a model that reaches near-parity at a small fraction of the reference's cost, and we argue below that this outcome is favourable for embedded deployment.

6.1 Trading width for precision

The accuracy curve is monotonic but diminishing, and the marginal benefit of each additional increment appears to shrink as the model grows, consistent with the generally diminishing returns of capacity scaling: once a binary model has enough channels to represent the dominant features, further width buys progressively less [14]. This suggests that model capacity is not the only thing binarization takes away.

The strongest evidence for this comes from the variant which retains the downsampling layers in full precision while keeping everything else binary. At the widest setting this recovers a further 1.1 points, reaching 87.8% and shrinking the gap to 1.1 points. This points to two distinct sources of the quantization gap.

One is a loss of representational capacity, which width scaling addresses directly. Here the route matters: the ablation of Section 5.3 shows that the intermediate and hidden dimensions recover capacity at very different accuracy-per-cost rates, underscoring that which dimension you widen matters, not just how much. The other source is a localized precision bottleneck at the downsampling layers, where the spatial resolution is halved and the channel count doubled. Binarizing these layers discards information at precisely the points where the representation is being compressed, and no amount of downstream width can recover information that was already destroyed upstream. Width scaling and selective precision therefore act on different parts of the problem, which explains why their gains are largely additive.

6.2 The training bottleneck

A central premise of the equal-budget framing in Equation 9 is that binarization frees memory and compute that can be reinvested into width. Our results show that, at least on this task, this budget is nowhere near exhausted. Even the widest fully binary model consumes only 13.7 MB and 382 MOPs, a $3.5\times$ and $6.7\times$ reduction relative to the reference, and the variant with full-precision downsampling layers still sits at $1.9\times$ and $2.4\times$ below budget.

The factor that actually limited how far we could push width was not the inference budget but the cost of training: because quantization-aware training maintains full-precision latent weights, a width-scaled binary model costs as much to train as a full-precision model of the same width, and the warm-start protocol adds a further full-precision model at every width. The equal-budget question is thus answered within a window set by training resources, rather than by the deployment constraint it was designed around, and the unspent inference budget leaves clear room for larger widths to be explored in future work.

6.3 Practical significance

A binary model within 1 to 2 points of full-precision accuracy at a fraction of the memory and compute is an attractive choice for power- and memory-constrained hardware. There are two caveats, however.

First, the compute and memory figures rest on the unified OPs metric, in which 64 bitwise operations count as a single floating-point operation (Section 2.2) [9]. To the best of our knowledge, no public inference framework ships the bitwise kernels needed to execute binary ViTs efficiently; therefore, these savings remain theoretical.

Second, our experiments use the small Oxford-IIIT Pet dataset, chosen to make a controlled multi-width sweep feasible with the available time and training hardware. The 7.1-point quantization gap we observe is markedly smaller than the 10.9 points reported for BHViT on ImageNet-1K (Table 1), so both the size of the gap and the ease with which width recovers it may be dataset-dependent. Whether the width-for-precision exchange behaves the same way at large scale, and whether the remaining gap closes once larger widths become trainable, remains open.

7 Conclusion

This work investigated whether the memory and compute freed by binarizing a Vision Transformer can be reinvested into model width to recover the accuracy lost to quantization, under a fixed compute and memory budget. Using BHViT-Tiny on the Oxford-IIIT Pet dataset, we measured a 7.1-point Top-1 drop attributable to binarization alone, then scaled width to recover it. Tripling the width raised the fully binary model from 81.8% to 86.7%, recovering 4.9 of the 7.1 points but leaving a 2.2-point gap, while keeping the downsampling layers in full precision recovered a further 1.1 points, narrowing the gap to 1.1 points. Every variant stayed far below budget: the widest using only 13.7 MB and 382 MOPs, a $3.5\times$ and $6.7\times$ reduction relative to the reference. These theoretical savings are not yet realized on hardware, however: lacking an optimized 1-bit kernel, we deployed the binary models under INT8 on a Raspberry Pi 5, where the narrow model improved modestly on latency and RAM over the full-precision baseline but the wide variant exceeded it on both. The reported latencies are thus an upper bound, and realizing the practical benefits of width scaling will require a dedicated binary inference framework. Overall, the results demonstrate that reinvesting quantization savings into width is an effective strategy for binary Vision Transformers, enabling models that approach full-precision accuracy while retaining substantial reductions in memory and computational cost.

8 Future Work

Larger widths on more capable hardware. Exploring larger widths on more capable training hardware would establish whether the width-for-precision exchange eventually reaches full parity or plateaus before it.

Large-scale validation. Validating the trade-off on a large-scale dataset where the quantization gap is known to be larger would test how well these conclusions generalize.

Mixed-precision layer assignment. The observation that selectively keeping the downsampling layers in full precision closes a disproportionate share of the residual gap motivates a more systematic study of mixed-precision layer assignment.

Hardware realization. Realizing these models on embedded hardware with optimized low-bit inference kernels would confirm the practical deployment benefit that this work establishes only in principle.

9 Responsible Research

9.1 Reproducibility

All experiments use the publicly available Oxford-IIIT Pet dataset [12] under its standard train/test split, and build on the official open-source BHViT implementation [5]. The full architecture and training configuration is reported in Appendix A, and our training and evaluation code, including configuration files for every width-scaled variant, is available at https://github.com/ivarvl/bhvit-width-scaling. All models were trained on a single RTX 5080 GPU.

9.2 Environmental Considerations

Quantization-aware training maintains full-precision latent weights, so training cost scales with width as for a full-precision model, and the warm-start protocol requires an additional full-precision run per width. The complete study comprised training runs totalling approximately 400 GPU-hours. This cost is incurred once at training time. The resulting binary models are intended to reduce energy consumption at inference, where a model's footprint typically accumulates.

9.3 Dataset and Ethics

The Oxford-IIIT Pet dataset contains images of animals only and no personally identifiable information. It is distributed for research use under a Creative Commons license. This classification task carries no foreseeable dual-use or fairness concerns.

9.4 Use of Large Language Models

Generative AI tools were used to improve the clarity and style of the writing of this paper. All AI-assisted content was carefully reviewed and edited by the author. The research design, implementation, experiments, and analysis were carried out entirely by the author.

A Training Hyperparameters

Table 5: Architecture and training configuration of the BHViT-Tiny architecture on the Oxford-IIIT Pet dataset. The same settings are used for all FP, binarized and width-scaled variants; only the base width d and the numerical precision differ. Training proceeds in two phases of 1,000 epochs each (a from-scratch Phase 1 (always FP) and a warm-started Phase 2, Section 4.2), both using the optimization settings below.

Parameter	**Value**
Architecture	
Per-stage embedding dims	$[64, 128, 256, 512]$
Per-stage intermediate sizes	$[512, 1024, 1024, 2048]$
Depths (blocks per stage)	$[2, 2, 6, 2]$
Attention heads per stage	$[1, 1, 4, 8]$
Input resolution	224×224
Patch size P	4
Window size	7
Stochastic depth (drop-path)	0.1
Number of classes	37
Optimization (per phase)	
Epochs per phase	$1{,}000$
Batch size	24
Optimizer	AdamW
Base learning rate	5×10^{-4}
LR schedule	Cosine
Warm-up epochs	5
Weight decay	0.05
Distillation teacher	DeiT-Tiny (91% Acc)
Data augmentation	
RandAugment	`rand-m9-mstd0.5-inc1`
Colour jitter	0.4
Random erasing prob.	0.25
Mixup α	0.8
CutMix α	1.0

B Raspberry Pi 5 Benchmark

Table 6: Inference latency and peak RAM usage on a Raspberry Pi 5, measured with ONNX Runtime. As no optimized binary (XNOR/popcount) inference kernel is available on this platform, the binary models are deployed with INT8 quantization, though the weights and activations remain binary.

Model	w (d)	RAM (MB)	Latency (ms)
FP Baseline	$1\times$ (64)	588	248
INT8	$1\times$ (64)	554	202
INT8	$3\times$ (192)	835	651

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR*, abs/1308.3432, 2013.

[2] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: practical automated data augmentation with a reduced search space. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[3] ONNX Runtime developers. ONNX Runtime. https://onnxruntime.ai/, 2021. Version: 1.26.0.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021.

[5] Tian Gao, Yu Zhang, Zhiyuan Zhang, Huajun Liu, Kaijie Yin, Chengzhong Xu, and Hui Kong. BHViT: Binarized Hybrid Vision Transformer. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3563–3572, 2025.

[6] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael Mahoney, and Kurt Keutzer. *A Survey of Quantization Methods for Efficient Neural Network Inference*, pages 291–326. Chapman and Hall/CRC, 2022.

[7] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the Knowledge in a Neural Network. *CoRR*, abs/1503.02531, 2015.

[8] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. ReActNet: Towards Precise Binary Neural Network with Generalized Activation Functions. In *European Conference on Computer Vision (ECCV)*, 2020.

[9] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-Real Net: Enhancing the Performance of 1-Bit CNNs with Improved Representational Capability and Advanced Training Algorithm. In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV*, pages 747–763, Berlin, Heidelberg, 2018. Springer-Verlag.

[10] Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits. *CoRR*, abs/2402.17764, 2024.

[11] Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. In *International Conference on Learning Representations*, 2020.

[12] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and Dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[13] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 525–542, Cham, 2016. Springer International Publishing.

[14] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019.

[15] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient Transformers: A Survey. *ACM Comput. Surv.*, 55(6), December 2022.

[16] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR, 18–24 Jul 2021.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[18] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith, editors, *BMVC*. BMVA Press, 2016.