When Are 1.58 Bits Enough? A Bottom-up Exploration of Quantization-Aware Training with Ternary Weights

Jacob Nielsen[a], Lukas Galke[b] and Peter Schneider-Kamp[c]
Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark

Keywords: Quantization, Quantization-Aware Training, Graph Neural Networks, Text Classification, Language Models.

Abstract: Contemporary machine learning models, such as language models, are powerful, but come with immense resource requirements both at training and inference time. Quantization aware pre-training with ternary weights (1.58 bits per weight) has shown promising results in decoder-only language models and facilitates memory-efficient inference. However, little is known about how quantization-aware training influences the training dynamics beyond such Transformer-based decoder-only language models. Here, we engage in a bottom-up exploration of quantization-aware training, starting with multi-layer perceptrons and graph neural networks. Then, we explore 1.58-bit training in other transformer-based language models: encoder-only and encoder-decoder models. Our results show that in all of these settings, 1.58-bit training is on par with standard 32/16-bit models, yet we also identify challenges specific to 1.58-bit encoder-decoder models. Our results on decoder-only language models hint at a possible regularization effect introduced by quantization-aware training.

1 INTRODUCTION

Large Language Models (LLMs) have dominated the field of natural language processing in recent years (Vaswani et al., 2017; Brown et al., 2020; Wei et al., 2022). However, LLMs come with deployments obstacles, especially regarding memory use, latency, and throughput and considerations regarding LLM's environmental impact in terms of energy consumption are threatening their uptake (Bommasani et al., 2021; Schwartz et al., 2020).

Post-training quantisation methods have been proposed, including but not limited to, Generative Pre-trained Transformer Quantization (Frantar et al., 2023) and Activation-aware Weight Quantization (Lin et al., 2024). Post-training quantization has also been employed in other domain such as computer vision (Li and Gu, 2023). However, post-training quantization inherently comes with a decrease in performance.

An alternative to post-training quantization is quantization-aware training such as LLM-QAT (Liu et al., 2023c) and QA-LoRA (Xu et al., 2023). Here, the quantized weights are subject to training such that there is no decrease in performance when the quantized model is then usedfor inference. Recent works on 1-bit (Wang et al., 2023) and 1.58-bit (Ma et al., 2024) quantization-aware training architectures have demonstrated the potential of training in a low-bit precision while maintaining most of the LLM's performance.

The 1.58-bit quantization-aware training architecture introduced in BitNet b1.58 (Ma et al., 2024) quantizes the weights of linear layers to the values -1, 0, or 1 during forward passes. It has been shown to yield only minimal performance decrease in 3B+ parameter LLMs. Building on this, recent work suggests that 1.58-bit training is promising for multimodal architectures (Sundaram and Iyer, 2024) and spiking language models (Bal et al., 2024). Investigating this b1.58 training scheme in small vision and language models (ranging from 100K to 48M parameters), Nielsen and Schneider-Kamp (2024) identify scaling laws between 16-bit and 1.58-bit training, showing that 1.58-bit can achieve similar performance to 16-bit training, even on these small lower-capacity networks. Furthermore, they show that employing the median (as a basis for quantization and rescaling) instead of the mean achieves even better performance in some settings.

However, b1.58 architectures and training schemes still pose numerous unanswered questions,

[a] https://orcid.org/0009-0009-8141-630X
[b] https://orcid.org/0000-0001-6124-1092
[c] https://orcid.org/0000-0003-4000-5570

Nielsen, J., Galke, L. and Schneider-Kamp, P.
When Are 1.58 Bits Enough? A Bottom-up Exploration of Quantization-Aware Training with Ternary Weights.
DOI: 10.5220/0013382400003890

In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 3*, pages 1440-1449
ISBN: 978-989-758-737-5; ISSN: 2184-433X

which require further investigation of its potential and limitations. Here, we investigate 1.58-bit quantization-aware training in a bottom-up manner, starting from the classic exclusive-or (X-OR) task and non-transformer architectures such as multi-layer perceptrons and graph neural networks. We further fill a gap in the literature by analyzing 1.58-bit quantization-aware training for encoder-only and encoder-decoder transformer-based language models, complementing prior work on decoder-only language models.

Our results suggest that 1.58-bit weights can be employed as a drop-in replacement for Linear layers in a multitude of model architectures with no to minimal loss in performance. We find that in encoder-only language models, commensurate accuracy with 16-bit models can be obtained by increasing the hidden size throughout the model. However, we also find that encoder-decoder transformer models, such as in 1.58-bit variants of T5 (Raffel et al., 2020), yield lower performance than their 16-bit counterparts.

In summary, our contributions are as follows:

- A bottom-up exploration of 1.58-bit quantization-aware training, ranging from the X-OR problem and 1.58-bit multi-layer perceptrons for text classification to 1.58-bit graph neural networks for node classification (Section 3).
- Experiments on 1.58-bit encoder-only language models showing that an increase in model capacity can compensate for the lower bit precision – yet sub-proportionally to the decrease in bit width (Section 4.1).
- Experiments on 1.58-bit encoder-decoder language models showing that these models suffer more from 1.58-bit training and struggle to reach the performance of 16-bit. (Section 4.2).

2 BACKGROUND: b1.58 QUANTIZATION

We recapitulate the basics of 1.58-bit quantization proposed by Nielsen and Schneider-Kamp (2024), which generalizes the one from (Wang et al., 2023).

The core of the 1.58-bit quantization scheme is to introduce a drop-in replacement for the `Linear` layers in common machine learning frameworks such as PyTorch, which we denote as `BitLinear`. During training, the BitLinear layer retains 16-bit shadow weights. During the forward pass, the shadow weights are quantized to 1.58-bit precision which corresponds to ternary weights: $\{-1,0,1\}$. During the backward pass, the shadow weights are optimized via the straight-through estimator (Bengio et al., 2013). Because forward passes are always conducted with quantized weights, we can drop the shadow weights when training concludes, using solely the ternary weights during inference.

The computation flow of a `BitLinear` layer follows a five-step procedure: First, the activations are normalized via a parameter-free LayerNorm (Ba et al., 2016), denoted here as $\hat{I}$ for input I .

Second, the layer normalized activations are quantized to k-bit precision (usually $k = 8$) via AbsMax quantization. We first calculate a scaling factor x_{scale} such that $x_{\text{scale}} = \frac{Q_b}{\max(|\hat{I}|)+\varepsilon}$, where $Q_b = 2^{k-1}$ is the range of the k bits used for the quantized activations and ε is a small positive value preventing zero-division. This ensures that all activations can be scaled to integer values $\{-Q_b, \ldots, Q_b - 1\}$. We then employ AbsMax quantization on the activations as follows:

$$\mathbf{x}_{\text{quant}} = \max(-Q_b, \min(Q_b - 1, \text{round}(\hat{\mathbf{I}} \cdot x_{\text{scale}}))$$

Third, the 16-bit shadow weights $W \in \mathcal{R}^{n \times m}$ are scaled and quantized to a ternary system of integer values in $\{-1,0,1\}$ (corresponding to 1.58 bits per weight through a generic AbsMeasure quantization method (Nielsen and Schneider-Kamp, 2024). For this, we calculate a second scaling factor $w_{\text{scale}} = \frac{1}{\text{Measure}(|W|)+\varepsilon}$, where Measure denotes either the mean or median function. The quantized weights $\mathbf{W}_{\text{quant}}$ can then be derived as:

$$\mathbf{W}_{\text{quant}} = \max(-1, \min(1, \text{round}(\mathbf{W} \cdot w_{\text{scale}}))$$

Fourth, having quantized both activations and weights, we can conduct a forward pass with $\mathbf{x}_{\text{quant}}$ and $\mathbf{W}_{\text{quant}}$:

$$\mathbf{y}_{\text{quant}} = \mathbf{x}_{\text{quant}} \cdot \mathbf{W}_{\text{quant}} + \mathbf{b}$$

where $\mathbf{b}$ is an optional bias term. To fully exploit the positive impacts on memory use, latency, and throughput, the forward pass ought to be carried out by a specialized kernel using bit-level operations.

We detach both $\mathbf{x}_{\text{quant}}$ and $\mathbf{w}_{\text{quant}}$ from the computation graph to facilitate a straight-through estimation of the gradients. The gradients will then be estimated with respect to the shadow weights, i.e., the 16-bit weights that were quantized via AbsMeasure in step 3.

Lastly, the result of this linear transformation is rescaled with the scaling factors x_{scale} and w_{scale} from steps 2 and 3, respectively. Thus, to calculate the layer's final output $\mathbf{y}$, we rescale y as follows:

$$\mathbf{y} = \frac{\mathbf{y}_{quant}}{w_{scale} \cdot x_{scale}}$$

3 BitNet IN NON-TRANSFORMER MODELS

Prior work on BitNet quantization has predominantly focused on analyzing transformer models (Wang et al., 2023; Ma et al., 2024; Nielsen and Schneider-Kamp, 2024). Here, we ask to what extent quantization-aware training is a feasible strategy for neural networks in general. We engage in a bottom-up exploration, starting from the popular X-OR problem and a minimalist BitNet model, then advance to BitNet in multilayer perceptrons to carry out text classification from a bag-of-words representation, and, finally, explore BitNet for graph neural networks.

3.1 Can 1.58-bit Models Solve X-OR?

To better understand the dynamics of 1.58-bit training, we explore the X-OR problem, in which the model needs to learn the function of exclusive-or: Given two binary inputs, the task is to output 1 when exactly one of the inputs is 1 and 0 otherwise. The X-OR-problem is particularly interesting because it is known that it cannot be solved by a purely linear model and requires at least one hidden layer with a non-linear activation.

In theory, ternary weights as in BitNet are sufficient to solve X-OR. One possible solution would be to have a two hidden units, one assigning a positive weight to the first input and a negative weight to the second, and vice-versa for the second hidden unit: $h_1 = \text{ReLU}(x_1 - x_2)$ and $h_2 = \text{ReLU}(x_2 - x_1)$. The output layer would then have two positive weights $y = h_1 + h_2$, solving the X-OR problem. Whether these weights can be learned from data is a question subject to experimentation.

Setup. We set up the basic X-OR-problem, while adding two extra noise inputs which do not affect the output and ought to be ignored (e.g., by assigning weight zero). We are interested whether a BitNet model with a single hidden layer, i.e., two BitLinear layers with an intermediate ReLU activation, can learn a perfect solution to the X-OR problem while ignoring the noise inputs. The weight range is set to 1.58 bits (-1, 0, or 1), the activation range is 8 bits. We train for 1k epochs on 5k examples with 4 features, two of which determine the target X-OR output. The learning rate is set to 0.01 unless noted otherwise. Optimization is carried out by Adam (Kingma and Ba, 2015) to minimize cross-entropy with the X-OR ground truth.

Results. A standard MLP with one hidden layer of two hidden units solves X-OR. BitNet variants with two hidden units do not find a solution.

With 8 hidden units, and the mean quantization scheme, AbsMean, the model finds a perfect solution (100% accuracy) with exactly 4 nonzero parameters on the input layer and, in particular, all zero weights on the noise inputs. However, with the median weight quantization (AbsMedian), the model did not find a solution with 8 hidden units, converging at accuracy 86.8%. The L1 norm of the X-OR input weights was 12 (out of 16) and the L1 norm of the noise input weights was 8 (out of 16).

When we increase the number of hidden units to 16, AbsMedian found a solution with perfect accuracy at the end of training, but the trajectory during training was unstable. The L1 norm for X-OR input weights was 24. The L1 norm for noise input weights was 17. Output layer L1 norm 26 (out of 32).

With 32 hidden units, AbsMedian found a perfect solution (100% accuracy) with less fluctuation on the trajectory. The L1 norm of the weights for X-OR inputs was 51 (out of 64). The L1 norm of the noise-input weights was 28 (out of 64). The L1 norm of the output layer was 50 (out of 64).

Going back to 8 hidden units, but larger learning rate (0.1), AbsMedian also finds a 100% accurate solution. The L1 norm of weights for X-OR inputs was 12 (out of 16), while the L1 norm of weights for noise inputs: 5 (4 of them negative). The hidden-to-output layer had all non-zero weights (L1 norm of 16).

The bias parameters seemingly help to ignore parts of the inputs together with ReLU activation. Note, when the weights for the noise inputs are negative, the bias term needs to compensate such that the noise inputs do not distort the sum with the X-OR inputs. Both AbsMean and AbsMedian managed to solve the X-OR problem. However, AbsMedian needed either a larger learning rate or a higher amount of hidden units.

3.2 BitNet in Multilayer Perceptrons

Next, we investigate to what extent a 1.58-bit multilayer perceptron model can learn to categorize texts based on bag-of-words features.

Setup. We use the same setup as a recent work on text classification (Galke and Scherp, 2022), where a wide multilayer perceptron (WideMLP) on a bag-of-words representation had shown good results. We train two BitNet variants (WideMLP-b1.58-mean and WideMLP-b1.58-median) of the WideMLP baseline. The first layer of the WideMLP model is implemented

Table 1: Text Classification: A wide multi-layer perceptron on a bag-of-words (WideMLP) compared to corresponding BitNet variants WideMLP-b1.58-mean and WideMLP-b1.58-median. WideMLP baseline results taken from Galke and Scherp (2022).

Method	20ng	R8	R20	ohsumed	MR	Average
WideMLP baseline	**83.31**	97.27	93.89	**63.95**	76.72	**83.03**
WideMLP-b1.58-mean lr=10^{-3}	79.89	**97.40**	93.54	60.75	**77.10**	81.74
WideMLP-b1.58-median lr=10^{-3}	80.08	97.35	**94.20**	62.28	76.14	82.01
WideMLP-b1.58-median lr=10^{-2}	81.74	96.80	93.69	62.73	76.22	82.24

as an embedding layer to avoid large matrix multiplications with the dimensions of the vocabulary size. Therefore, we quantize only the hidden-to-output layer of the two-layer architecture. Nevertheless, this leads to the interesting question of how quantized fully-connected layers cope with non-quantized embedding layers. We use 5 standard benchmark datasets: 20ng, R8, R52, ohsumed, and MR – four being topic classification and MR being sentiment classification. As in WideMLP, we train for 100 epoch with batch size 16.

Results. As shown in Table 1, WideMLP-b1.58-mean achieves 98.4% of WideMLP's performance (unweighted average over datasets). WideMLP-b1.58-median achieves 98.8% of WideMLP performance. With a learning rate of 10^{-2}, WideMLP-b1.58-median attains 99.0% of the WideMLP baseline performance.

3.3 BitNet in Graph Neural Networks

We seek to understand how BitNet affects graph representation learning. We evaluate 1.58-bit variants of graph convolutional networks (GCN; Kipf and Welling, 2016) and simplified graph convolution (SGC; Wu et al., 2019) on three commonly used citation datasets to evaluate node classification in graphs: Cora, Citeseer, Pubmed under the split by Yang et al. (2016): using only the labels of 20 nodes per class for training.

Setup. As base graph neural network models, we use a 2-layer ReLU-activated GCN and an SGC model with 2-hop neighborhood aggregation. In both models, we substitute the linear layers with BitLinear (2 for the GCN, and 1 for SGC). We then experiment with mean and median weight quantization. All models are trained for 100 epochs with learning rate 0.01. We report mean accuracy and 95% confidence intervals across 10 repetitions.

Results. Table 2 shows the results. For both GCN and SGC, the b1.58-mean and b1.58-median variants yield accuracy scores very close to their full-precision counterparts and even higher accuracy in some cases: SGC-b1.58-median on Cora and GCN-b.158-mean on Citeseer. On the unweighted averaged over all three datasets, GCN-b1.58-median achieves 98.8% relative performance to GCN and SGC-b1.58-median achieves 98.5% of SGC's performance.

3.4 Summary and Interim Discussion

What can we learn from this set of experiments? In the toy X-OR setting, we found that BitNet models need a higher number of parameters, or a higher learning rate – as also suggested by prior work on BitNet (Wang et al., 2023). However, in practical settings, such as text classification and node classification, BitNet variants yield almost the same performance as their respective baselines – even without increasing the learning rate or the number of model parameters. We hypothesize that this is due to sufficient overparameterizion.

The experiments with the SGC-b1.58 models show higher standard deviations, when compared to the GCN-b1.58 models. A potential reason is that SGC employs a linear mapping from neighborhood-aggregated features to classes, which could lead to higher variability as the model has less opportunities to compensate for the reduced precision.

As for the difference between mean and median quantization schemes, we find that both options lead to similar performance. In the following, we will proceed with the median option only.

4 BitNet IN TRANSFORMERS

We now move to transformer-based language models including encoder-only, encoder-decoder, and decoder-only transformer architectures employing BitLinear layers with the AbsMedian quantization for 1.58-bit weights and AbsMax quantization for 8-bit

Table 2: Node classification. Mean accuracy over 10 runs with 95% confidence intervals. Graph convolutional networks (GCN) and Simplified Graph Convolution (SGC) compared to their BitNet variants (*-b1.58-mean and *-b1.58-median).

Method	Cora	Citeseer	Pubmed	Avg.
GCN baseline	**78.57** $\pm$ 0.49	63.76 $\pm$ 0.48	**76.02** $\pm$ 0.19	**72.78**
GCN-b1.58-mean	76.03 $\pm$ 0.50	**65.83** $\pm$ 0.45	73.51 $\pm$ 0.47	71.79
GCN-b1.58-median	75.76 $\pm$ 0.32	65.60 $\pm$ 0.65	74.42 $\pm$ 0.39	71.93
SGC baseline	77.07 $\pm$ 0.15	**63.66** $\pm$ 0.18	**75.63** $\pm$ 0.08	**72.12**
SGC-b1.58-mean	77.31 $\pm$ 0.38	59.31 $\pm$ 1.82	75.22 $\pm$ 0.33	70.61
SGC-b1.58-median	**77.46** $\pm$ 0.75	61.31 $\pm$ 1.90	74.42 $\pm$ 0.26	71.06

activations. We will show that the performance of b1.58 in encoder-only architectures align with previous results on decoder-only architectures, but that this does not hold for the encoder-decoder architecture.

4.1 Encoder-Only Language Models

Here, we experiment with 1.58-bit variants of the encoder-only model BERT (Devlin et al., 2019).

Setup. We employ the Cramming framework (Geiping and Goldstein, 2023) and the crammed-BERT architecture[1] with the bert-o4 config[2] provided by the framework to standardize our experiments. As such, we use a learning rate of 10^{-3}, `AdamW`-optimizer (Loshchilov, 2017) and a batch size of 8192. We train for masked language model objective with the default masking probability of 25%. We experiment with different hidden sizes for the 16-bit and 1.58-bit models.

Motivated by low-resource language modeling, the experiments are conducted with a dataset consisting of approximately 80% Danish texts mixed with 20% equal amounts of Norwegian, Swedish, German, and English over 6.2 million documents, tokenized into 517M tokens with BERT tokenizer (Wu et al., 2016), and into 403M tokens for T5's tokenizer (Kudo, 2018). The dataset is collected from high quality text sources and split into train and validation sets balanced equally from data sources. Due to copyright restrictions and usage agreements, the dataset cannot be published.

Results. We compare the performance of 16-bit and 1.58-bit training in Figures 1a and 1b. Encoders scale as outlined by prior work, needing approximately twice the hidden layer size to achieve performance comparable to 16-bit versions. For instance, a hidden layer size of 384 in b1.58 archives performance similar to hidden size of 192 in 16-bit. This trend is observed across other pairs of hidden sizes, e.g., 768 and 384 as well as 1536 and 768.

4.2 Encoder-Decoder Language Models

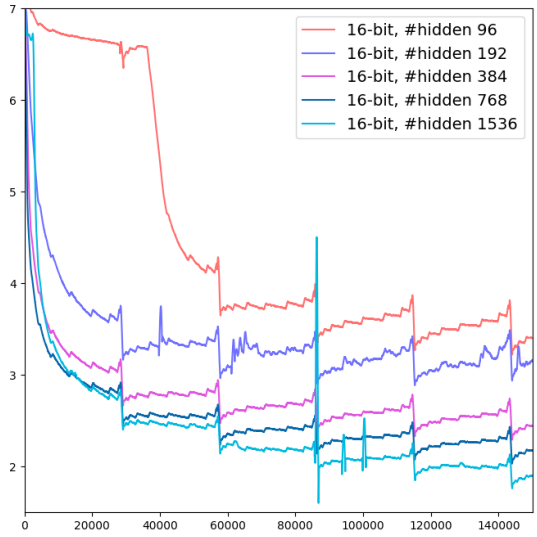
Next, we investigate the impact of 1.58-bit training on T5 (Raffel et al., 2020) as an encoder-decoder model.

Setup. We apply b1.58 to the standard masked span reconstruction objective with a masking probability of 15%. We use the NanoT5 framework (Nawrot, 2023) with a `T5v1_1-base` architecture and `mT5`'s SentencePiece tokenizer (Kudo, 2018). Specifically, we use a learning rate of 0.02, a batch size of 128, an input sequence length of 512, a `cosine` learning rate scheduler and the `adamwscale` optimizer (Nawrot, 2023). The architecture has 6 layers in the encoder and another 6 in the decoder stack (Fig. 2). Hidden sizes are varied between 48 and 1536. Experiments are conducted on the dataset described in Section 4.1.

Results. We compare the 16 bit and 1.58 bit performance in Figure 2. In Figure 2a we compare different hidden size's performance between 16-bit and 1.58-bit (median). We observe that every 16-bit version (solid line) outperform all the 1.58-bit versions (dashed line). We refer to Figure 2b evaluating the models on held-out validation set, however, coming from the same distribution. We observe that the validation loss aligns almost perfectly with their corresponding training loss curve, likely indicative of decent in-distribution generalization and decent downstream task performance (Liu et al., 2023a). In general, we observe the 1.58-bit versions to exhibit more unstable training behavior, resulting in an increase in loss over time, which is also reflected in the validation performance.

[1] https://github.com/JonasGeiping/cramming/blob/main/cramming/config/arch/crammed-bert.yaml

[2] https://github.com/JonasGeiping/cramming/blob/main/cramming/config/train/bert-o4.yaml



(a) Scaling for 16 bit

(b) Scaling for 1.58 bit (median)

Figure 1: Scaling Behavior of 16-bit and 1.58-bit (median) BERT models. Training loss over 150K optimization steps. Smoothing applied with a Savitzky-Golay filter with a window size of 1000 and a polynomial order of 2.

4.3 Decoder-Only Language Models: Quantization as a Regularizer?

Here, we experiment with decoder-only language models to better understand the training dynamics. We hypothesize that there could be a regularization effect of 1.58-bit quantization due to coarser representations.

Setup. To investigate this potential regularization effect, we experiment with Open Language Models (OLMo Groeneveld et al., 2024) with 1B parameters. We employ b1.58 with weight-only quantization using AbsMedian with a 1.58-bit weight range. We train on the dataset described in Section 4.1 with OLMo's standard hyperparameters [3], which include a sequence length of 2048 and a batch size of 2048.

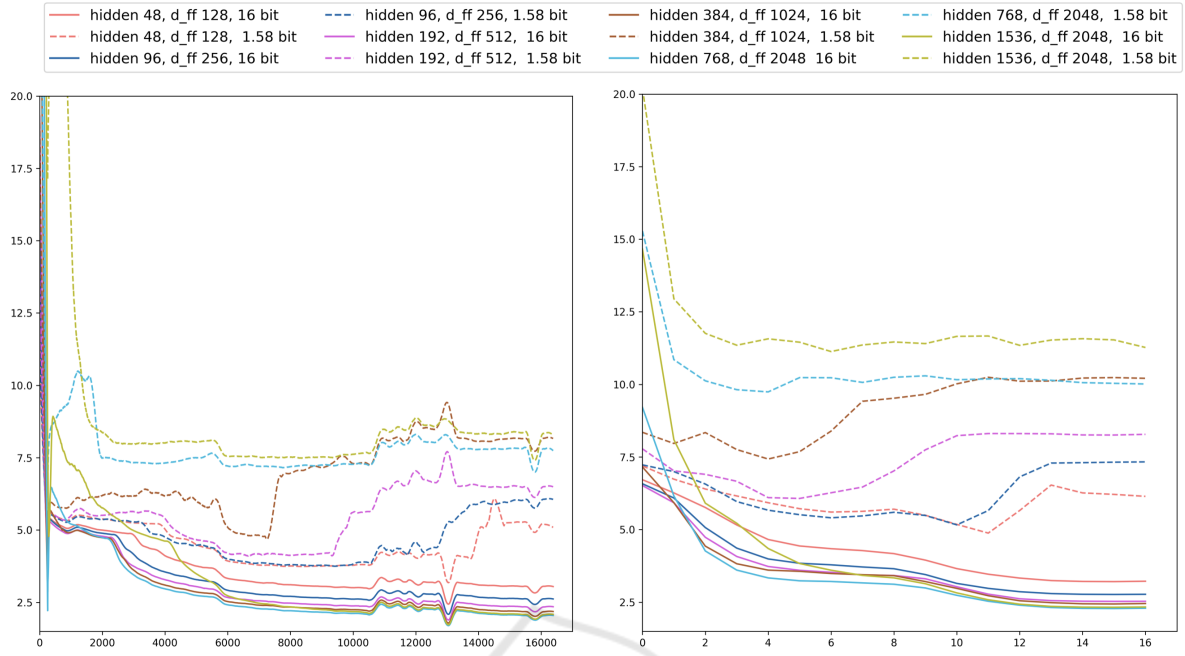
Results. Figure 3 shows the results of the OLMo models. As expected, we observe that 16-bit performs best (see Figure 3a). The b1.58 variant is worse at fitting the data early on and consistently shows only a small decrease in loss. However, in Figure 3b, we see that b1.58 delays overfitting to the training dataset. We attribute this to a regularization effect the quantization must introduce, making the b1.58 versions superior on the validation set. We see that 1.58-bit without and with dropout perform better and best, respectively, on the validation set. These results indicate that the coarseness of the ternary quantization also contributes positively to model regularization (see Figure 3) and a good generalization performance.

[3]https://github.com/allenai/OLMo/blob/main/configs/official/OLMo-1B.yaml

5 DISCUSSION

We have shown that BitNet models can generally be trained to yield commensurate accuracy with their standard 16/32-bit counterparts in multi-layer perceptrons, graph neural networks, encoder-only language models, and decoder-only language models – only T5-like encoder-decoder architectures pose extra challenges. Our results with decoder-only language models further hint at a possible regularization effect being introduced by quantization-aware training. Together, these results highlight that 1.58-bit quantization-aware training methods could be applied more widely and thereby greatly reduce memory requirements at inference time for a wide range of models.

Our findings are particularly interesting in the light of very recent considerations on scaling laws in the context of quantization (Kumar et al., 2024; Nielsen and Schneider-Kamp, 2024). Kumar et al. (2024) propose *scaling laws for precision*, claiming that to some extent bit-precision and parameter count balance each other out and that the compute-optimal bit-width lies around 7-8 bits. Nielsen and Schneider-Kamp (2024) instead argue that the need to increase parameters is sub-proportional to the reduction in memory size attained by 1.58-bit quantization-



(a) Training Loss. `[savitzky-golay, 512, 2]`

(b) Validation Loss. `[savitzky-golay, 4, 2]`

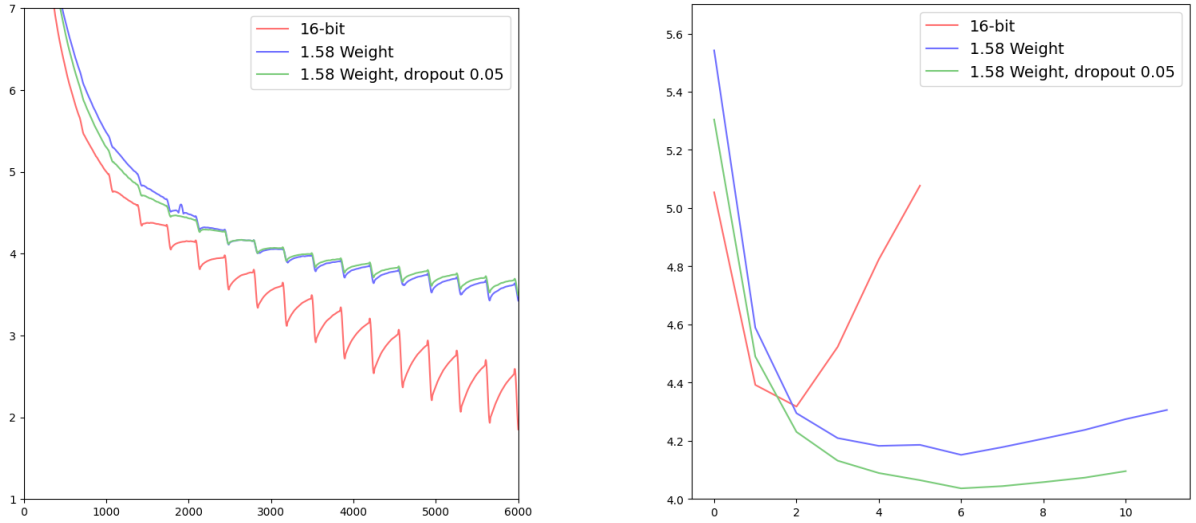
Figure 2: Scaling behavior of a nanoT5 encoder-decoder language model (T5v1.1-base), comparing 16 bit with 1.58-bit quantization-aware training with quantization applied to throughout the entire T5 model. `d_ff` denotes the hidden size of the feedforward components within each encoder and decoder stack.

aware training. Our current study contributes to these discussion by showing that in several cases beyond language models, the parameter count does not even need to be increased, and most importantly, we have observed a regularization effect for decoder-only language models – a highly interesting phenomenon of quantization-aware training if confirmed in future work.

In more detail, we have investigated how encoder-only transformers scale with 1.58-bit quantization aware-training. Analyzing the scaling behaivour of 16-bit and 1.58-bit models, we observed the need to re-introduce capacity in some cases. This is inline by the considerations of both Kumar et al. (2024) and Nielsen and Schneider-Kamp (2024). Specifically, we see the need to use a hidden size of 192 in 1.58-median to gain the same performance as 16-bit with a hidden size of 96. This also holds for hidden layer sizes of 384, 768 and 1536 for 1.58-median paired with 192, 368 and 768 for 16-bit, respectively. This confirms that, there is a subproportional scaling law at play for quantization-aware training of encoder-only transformers – similar to previous observations on decoder-only language models by Nielsen and Schneider-Kamp (2024),

In Figure 2a, we have observed ed “knees” forming in multiple loss curves, where the training exhibits a sudden drop in loss at a certain point of time, e.g. in the teal, blue, red and pink graph (dashed and solid). Both the 16-bit and 1.58-bit versions exhibit this behavior, where the 16-bit versions dropping around 2200-3000 step mark, where the 1.58-versions are delayed and in general yield a less significantly and spread out drop. Similar phenomena have been observed by Chen et al. (2024) on masked language models, who was found that this is the point where the models picks up the syntactic structure of the language. We presume that the knees in the loss curve we observe here also hint at syntax acquisition, which is important to confirm for quantization-aware training.

We have further shown that findings for the encoder-decoder transformer architectures do not directly align with the findings for the separate encoder-only and decoder-only architectures. Employing 1.58-bit throughout the model generally degrades performance substantially, as observed both on the training and the validation loss – and this cannot be compensated by scale as to the limits of our experiments. Note that these results do not include a full epoch worth of optimization steps. Therefore, one cannot attribute this effect to the model seeing the same exemplars multiple times. Notably, even an increased hidden size, did not lead to the 1.58-bit models catching up with 16-bit models, as was the case in all other settings studied in this paper. We hypothesize that this may be caused by the specific setup of cross- atten-



(a) Training Loss

(b) Validation Loss

Figure 3: Regularization Effect of b1.58 (median), OLMo 1B model trained on the dataset described in Section 4.1. Smoothing is applied using a Savitzky-Golay filter with a window size of 1000 and polynomial order 2.

tion in the encoder-decoder architectures and encourage future work to investigate this effect further.

Comparing the two weight quantization schemes for BitNet: AbsMedian and AbsMean, we have observed that AbsMedian is on par with AbsMean. Prior work has shown that AbsMedian performs better in some situations (Nielsen and Schneider-Kamp, 2024), conjecturing AbsMedian to be more resilient to weight-updates, which allows for higher variance without noticeable effect on the scaling factor. While this may be a factor for models with millions of parameters, the miniature models involved in solving the X-OR task seem to be extraordinarily sensitive when the scaling factor is computed on a much smaller sample. We conjecture that this is the source of the instability observed in some configurations (e.g, hidden size 8 and 16 with low learning rate), which we have observed to be dampened when increasing the hidden size or the learning rate. In practical tasks, AbsMean and AbsMedian quantization are close to each other with a 0.6% drop in accuracy for MLPs and within each other's confidence intervals for GNNs.

We hypothesize that these previously described effects can be attributed to neural networks often not utilizing all parameters effectively and contain redundant parameters or even layers (e.g., see Ashkboos et al., 2024; He et al., 2024). Therefore, these existing parameters can be considered as surplus capacity when 1.58-bit quantization-aware training is employed, which would explain the smaller performance gap when the model size increases.

To fully harvest the potential of 1.58-bit quantization regarding memory use, latency, and throughput, there is a need for specialized kernels. Moreover, quantization-aware training for b1.58 comes with a slight increase in training resources due to having to quantize and scale weights and activations. However, in exchange, required resources at inference time are greatly reduced. Thus, together with specialized kernels, b1.58 models would contribute to more sustainable inference and serving of large models. Reducing the memory footprint also allows researchers and practitioners to carry out more computation locally and therefore may help alleviate privacy concerns.

Limitations. Due to to the high number of covered experimental settings, the experiments on language models are limited to analyzing loss trajectories: training loss for encoder-only models, and training and validation loss for encoder-decoder as well as for decoder-only models. We acknowledge that this does limit the generalizability of our results. Although the validation loss has been shown to be highly correlated with downstream performance in language models, there might be effects of quantization that could harm downstream performance of 1.58-bit models (Liu et al., 2023b).

6 CONCLUSION

We conducted a bottom-up investigation of 1.58-bit quantization-aware training for a range of both non-transformer and transformer models, demonstrating

very competitive performance for multi-layer perceptrons, graph neural networks, encoder-only, and encoder-decoder architectures when compared to 16 and 32-bit. Specifically, we find that b1.58-bit training works well beyond language models, demonstrating competitive performance in bag-of-words MLPs for text classification and graph neural networks for node classification. Median quantization seems on par or better than mean in most real-world scenarios. We further analyze the "BitNet Scaling Law" for encoder-only models, showing that 1.58-bit models match the training performance of standard precision models when the hidden size is twice as large, aligning with similar observations for decoder-only models. For encoder-decoder models, we find that no such scaling law is applicable, as b1.58 consistently performs worse than 16-bit. We encourage future work to investigate the challenges of b1.58 quantization with encoder-decoder architecture in more depth. Finally, there seems to be a regularization effects of 1.58-bit quantization-aware training that helps generalization. Yet, more research is needed to further investigate this regularization effect.

ACKNOWLEDGEMENTS

We are grateful to the Danish Foundational Models (DFM) project for access to data for low-resource language modelling, to SDU UCloud and EuroHPC Leonard Booster for providing computational resources.

REFERENCES

Ashkboos, S., Croci, M. L., do Nascimento, M. G., Hoefler, T., and Hensman, J. (2024). Slicegpt: Compress large language models by deleting rows and columns.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bal, M., Jiang, Y., and Sengupta, A. (2024). Exploring extreme quantization in spiking language models. *arXiv preprint arXiv:2405.02543*.

Bengio, Y., Léonard, N., and Courville, A. C. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R. B., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N. S., Chen, A. S., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L. E., Goel, K., Goodman, N. D., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M. S., Krishna, R., Kuditipudi, R., and et al. (2021). On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33*.

Chen, A., Shwartz-Ziv, R., Cho, K., Leavitt, M. L., and Saphra, N. (2024). Sudden drops in the loss: Syntax acquisition, phase transitions, and simplicity bias in MLMs. In *The Twelfth International Conference on Learning Representations*.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. (2023). Gptq: Accurate post-training quantization for generative pre-trained transformers.

Galke, L. and Scherp, A. (2022). Bag-of-words vs. graph vs. sequence in text classification: Questioning the necessity of text-graphs and the surprising strength of a wide MLP. In Muresan, S., Nakov, P., and Villavicencio, A., editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4038–4051, Dublin, Ireland. Association for Computational Linguistics.

Geiping, J. and Goldstein, T. (2023). Cramming: Training a language model on a single gpu in one day. In *International Conference on Machine Learning*, pages 11117–11143. PMLR.

Groeneveld, D., Beltagy, I., Walsh, P., Bhagia, A., Kinney, R., Tafjord, O., Jha, A., Ivison, H., Magnusson, I., Wang, Y., Arora, S., Atkinson, D., Authur, R., Chandu, K. R., Cohan, A., Dumas, J., Elazar, Y., Gu, Y., Hessel, J., Khot, T., Merrill, W., Morrison, J. D., Muennighoff, N., Naik, A., Nam, C., Peters, M. E., Pyatkin, V., Ravichander, A., Schwenk, D., Shah, S., Smith, W., Strubell, E., Subramani, N., Wortsman, M., Dasigi, P., Lambert, N., Richardson, K., Zettlemoyer, L., Dodge, J., Lo, K., Soldaini, L., Smith, N. A., and Hajishirzi, H. (2024). Olmo: Accelerating the science of language models. *arXiv preprint*.

He, S., Sun, G., Shen, Z., and Li, A. (2024). What matters in transformers? not all attention is needed.

Kingma, D. P. and Ba, J. (2015). Adam: A method for

stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kudo, T. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Kumar, T., Ankner, Z., Spector, B. F., Bordelon, B., Muennighoff, N., Paul, M., Pehlevan, C., Ré, C., and Raghunathan, A. (2024). Scaling laws for precision. *arXiv preprint arXiv:2411.04330*.

Li, Z. and Gu, Q. (2023). I-vit: integer-only quantization for efficient vision transformer inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17065–17075.

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., and Han, S. (2024). Awq: Activation-aware weight quantization for llm compression and acceleration.

Liu, H., Xie, S. M., Li, Z., and Ma, T. (2023a). Same pretraining loss, better downstream: Implicit bias matters for language models. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 22188–22214. PMLR.

Liu, H., Xie, S. M., Li, Z., and Ma, T. (2023b). Same pretraining loss, better downstream: Implicit bias matters for language models. In *ICML*, volume 202, pages 22188–22214. PMLR.

Liu, Z., Oguz, B., Zhao, C., Chang, E., Stock, P., Mehdad, Y., Shi, Y., Krishnamoorthi, R., and Chandra, V. (2023c). Llm-qat: Data-free quantization aware training for large language models.

Loshchilov, I. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Ma, S., Wang, H., Ma, L., Wang, L., Wang, W., Huang, S., Dong, L., Wang, R., Xue, J., and Wei, F. (2024). The era of 1-bit llms: All large language models are in 1.58 bits.

Nawrot, P. (2023). nanoT5: Fast & simple pre-training and fine-tuning of T5 models with limited resources. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*. Association for Computational Linguistics.

Nielsen, J. and Schneider-Kamp, P. (2024). Bitnet b1. 58 reloaded: State-of-the-art performance also on smaller networks. In *International Conference on Deep Learning Theory and Applications*, pages 301–315. Springer.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O. (2020). Green AI. *Commun. ACM*, 63(12):54–63.

Sundaram, J. and Iyer, R. (2024). Llavaolmobitnet1b: Ternary llm goes multimodal! *arXiv preprint arXiv:2408.13402*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Wang, H., Ma, S., Dong, L., Huang, S., Wang, H., Ma, L., Yang, F., Wang, R., Wu, Y., and Wei, F. (2023). Bitnet: Scaling 1-bit transformers for large language models.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. (2022). Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019). Simplifying graph convolutional networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Łukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation.

Xu, Y., Xie, L., Gu, X., Chen, X., Chang, H., Zhang, H., Chen, Z., Zhang, X., and Tian, Q. (2023). Qa-lora: Quantization-aware low-rank adaptation of large language models.

Yang, Z., Cohen, W., and Salakhudinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 40–48, New York, New York, USA. PMLR.