Continual Quantization-Aware Pre-Training: When to transition from 16-bit to 1.58-bit pre-training for BitNet language models?

Jacob Nielsen and Peter Schneider-Kamp and Lukas Galke
Department of Mathematics and Computer Science
University of Southern Denmark
{jacn,petersk,galke}@imada.sdu.dk

Abstract

Large language models (LLMs) require immense resources for training and inference. Quantization, a technique that reduces the precision of model parameters, offers a promising solution for improving LLM efficiency and sustainability. While post-training quantization methods typically achieve 4-8 bits per parameter, recent research suggests that training LLMs with 1.58 bits per weight parameter from scratch can maintain model accuracy while greatly reducing memory requirements and energy consumption at inference time. Here, we investigate a training strategy for quantization-aware pre-training, where the models are first trained with 16-bit precision and then transition into 1.58-bit quantization-aware training. Our results on 11 downstream tasks show that this 16-to-1.58-bit training strategy is preferable over full 1.58-bit training and leaves models closer to those which have undergone 16-bit training. We further investigate the effects of retaining the optimizer state at the transition point and gradually phasing in quantization strength – finding that both techniques alleviate the magnitude of loss spikes, but also that these effects can be compensated through further training.

1 Introduction

Large Language Models (LLMs) have revolutionized natural language processing and are making a strong entry into both related and unrelated industries. However, deployment of LLMs comes with a series of obstacles such as memory-usage, latency, and throughput. Environmental considerations regarding energy consumption of both training and inference becomes an increasingly important aspect (Schwartz et al., 2020; Strubell et al., 2020).

Quantization-aware training of language models, i.e. preparing the model for later quantization already during training and thereby preventing performance degradation between training and inference, has shown promising results (Wang et al., 2023; Ma et al., 2024). However, the resulting models tend to require more parameters to compensate for the reduction in bit-precision per parameter (Kumar et al., 2025; Nielsen and Schneider-Kamp, 2024). Here, we investigate a strategy of first training with standard precision in an initial phase, followed by a second phase of 1.58-bit quantization-aware training with weights quantized to either -1, 0, or 1.

Reducing inference compute demands of language models is particularly critical in the context of recent advances in scaling test time compute (Guo et al., 2025; Muennighoff et al., 2025; Jaech et al., 2024). If the trend of scaling test-time compute continues, we can assume that inference will soon dominate the resource consumption in a language model's life cycle.

Recent works in 1-bit (Wang et al., 2023) and 1.58-bit quantization-aware training (Ma et al., 2024; Nielsen and Schneider-Kamp, 2024; Nielsen et al., 2024), demonstrate the potential of training in 1.58-bit precision while retaining most of the performance, mitigating some of the drawback of existing post-training quantization techniques, such as performance degradation in both NLP and computer vision (Frantar et al., 2023; Li and Gu, 2023). To achieve competitive model performance, these quantization-aware training strategies keep 16-bit-precision weights at training time ("shadow weights"), which are quantized on-the-fly to 1-bit or 1.58-bit precision during forward passes. Straight-through estimated gradients (Bengio et al., 2013) are then used to update the shadow weights.

While such a training strategy initially requires more memory and compute than pre-training a regular language model, its benefits can be harvested after training. At inference time, the final shadow weights can be quantized once and for all, after which the shadow weights can be discarded, yielding a model with 4-5 times reduced memory footprint, compared to 16-bit model. With tailored kernels, this would allows us to replace costly matrix multiplications within linear layers with computationally more efficient integer addition operations.

Although these methods provide an exciting avenue for efficient inference on models trained with 1.58-bit quantization-aware training techniques, existing models are not eligible for efficient inference out of the box (Wang et al., 2023). This is increasingly important as increasing model size implies corresponding resource requirements, making training from scratch both resource heavy and environmentally challenging. This calls for an investigation of a resource efficient strategy.

Recent analysis have hinted at 7-8 bits being compute optimal (Kumar et al., 2025). Pushing the bit-representation even lower currently requires a quantization-aware training strategy as proposed in BitNet (Wang et al., 2023; Ma et al., 2024).

Concurrent work on quantization-aware training has shown that it is generally possible to convert a 16-bit model and into a 1.58-bit model through continual pre-training (Mekkouri et al., 2024). However, the experiments reported upon only consider the case of starting with an already pre-trained language model and do not compare against training with low precision from scratch. In related matter, recent work has shown that "overtrained" networks, such as fully-pre-trained language models, do not represent an ideal starting point for quantization (Kumar et al., 2025).

It is so far unclear which strategy to choose when pre-training from scratch: Is it preferable to train with 1.58-bit quantization-aware training all the way, or is it preferable to start with standard 16-bit training and then transition into 1.58-bit quantization-aware training (see Figure 1). And if so, when should we switch from 16-bit to 1.58-bit training? Is a it beneficial to gradually phase in the quantization strength? And what role does the retention of the optimizer state (if available) play?

In this work, we investigate these questions to shed new light on the potential of continuing a pre-training run in 1.58-bit. We aim to provide insights into the potential of converting existing pre-trained models to 1.58-bits and into developing more efficient ways of pre-training new models. Ultimately, this work contributes to a future with more efficient, environmentally-friendly LLMs and also allows to convert existing 16-bit models into 1.58-bit models, even in low-resource settings.

In order to answer these questions and achieve our aims, we systematically perform a number of experiments to identify a data-optimal transition

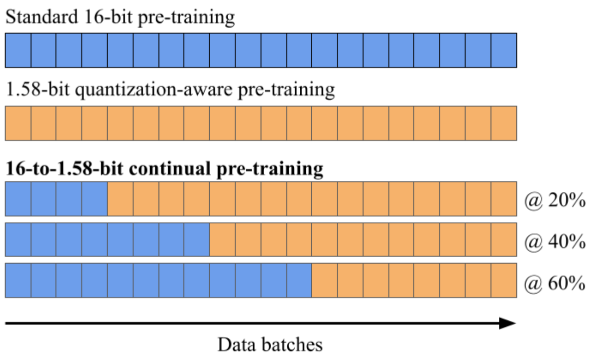

Standard 16-bit pre-training
1.58-bit quantization-aware pre-training
16-to-1.58-bit continual pre-training
@ 20%
@ 40%
@ 60%
Data batches

Figure 1: 16-to-1.58-bit continual pre-training. Blue and yellow denotes batches processed under 16-bit and 1.58-bit training, respectively.

point from 16-bit to 1.58-bit training. We further investigate whether it makes sense to phase in the quantization strength and whether the optimizer state is of importance, providing a basis for future work on expand upon and for practitioners to receive guidance in converting existing models and pre-training new models.

Challenging both conventional wisdom and common-sense expectations, our systematic comparison reveals that models that are first trained with 16-bit precision, and only later transition into 1.58-bit quantization-aware training, are more ultimately effective than models that are fully trained with 1.58-bit quantization-aware training.

In addition, we find that, while retaining the optimizer state indeed helps to alleviate a temporary increase in training loss, continual pre-training with a fresh optimizer state attains similar loss values after a limited number of optimization steps.

Furthermore, we find that gradually introducing quantization strength provides little to no benefits. A continual pre-training with full quantization strength temporarily increases training loss, but attains similar loss values, and also downstream performance, after a limited number of steps.

These results together indicate that the availability of optimizer state is not an obstacle when continuing pre-training from openly available language models and that no new hyperparameters (regarding phasing in quantization strength) are needed, enabling an easier transition of existing pre-trained models, with smaller amounts of data. In summary, our contributions are as follows:

- Systematic investigation and comparison between 1.58-bit pre-training from scratch and hybrid 16-bit training with 1.58-bit continual pre-training.

- Results from 11 downstream tasks showing that 16-to-1.58-bit continual pre-training is more effective than full 1.58-bit training.
- An analysis revealing limited effects of the gradual phasing in of quantization strength and the retention of optimizer states.

2 Related work

Early attempts of ternary-weight neural nets. The exploration into ternary weights was motivated by finding a better trade-off between accuracy and complexity than binary neural networks, which had suffered a substantial decrease in performance, effectively hindering the usability of such networks (Chen et al., 2021). Earlier attempts of binary- and ternary-weight neural networks showcased ternary superiority over binary weights, while showing promising results in the computer vision domain employing a direct optimization of the quantization (Li et al., 2016; Zhu et al., 2016).

Post-training quantization. The most common approaches for quantization fall under the category of post-training quantization. Those include approaches such as Generative Pre-trained Transformer Quantization (GPTQ) (Frantar et al., 2023) and Activation-aware Weight Quantization (AWQ) (Lin et al., 2024). A similar proposal for the vision transformer (Li and Gu, 2023) represents one of many efforts in the computer vision domain. Post-training quantization approaches come with an inherent decrease in performance, trading increased latency and throughput and decrease memory for precision (Kumar et al., 2025).

Quantization-aware training. Quantization-aware training was already proposed in earlier work on post-training such as LLM-QAT (Liu et al., 2023) and QA-LoRA (Xu et al., 2023). These methods directly optimize the quantized weights with respect to an objective function such that there is no decrease in performance when the model is used for inference.

Recently, we have seen a number of works on 1-bit (Wang et al., 2023) and 1.58-bit (Ma et al., 2024) quantization-aware techniques demonstrating strong performance in LLM performance, yielding a small or no loss in precision depending on model sizes. Other works have demonstrated strong potential for multi-modal architectures (Sundaram and Iyer, 2024) and spiking language models (Bal et al., 2024). Lastly, we have seen an investigation into the potential of 1.58-bit in small language and vision models and the definition of a scaling law for decoder-only models for reintroducing capacity (Nielsen and Schneider-Kamp, 2024). Similar scaling laws hold for encoder-only models while encoder-decoder models seem to be less predictable (Nielsen et al., 2024). This latest work further shows that also non-transformer models, such as plain multi-layer perceptions and graph neural networks can attain similar performance as their 16/32-bit counterparts, even without increasing the number of parameters.

Summary. Research on language model quantization shows promising results but is also limited by the effectiveness of the final models compared to standard precision models. There is a strict distinction between post-training methods and quantization-aware training. So far, it remains unexplored whether an initial phase of standard 16-bit precision training would improve or worsen the performance of the final model when continuing with 1.58-bit quantization-aware pre-training.

3 Methods

We first recapitulate the basics of quantization-aware training (Section 3.1) before describing the proposed strategy of continual 1.58-bit pre-training (Section 3.2) and discussing critical considerations concerning optimizer states (Section 3.3) and gradually phasing in quantization strength (Section 3.4).

3.1 Background on 1.58-bit training

Recent work has focused on specifically on 1.58-bit quantization-aware training techniques (Wang et al., 2023; Nielsen and Schneider-Kamp, 2024; Nielsen et al., 2024). Specifically, these works investigate ternary networks, where the weights only can take on the values -1, 0 and 1. The quantization is guided by 16-bit precision “shadow weights”, which are quantized during the forward passes and rely on a straight-through estimator for optimization. Activations are also commonly quantized in a given range Q_b, which is then multiplied with the quantized weight-matrix. Intuitively, the the weight-matrix then either is adding, ignoring, or subtracting the previous layer’s activations using -1, 0, or 1, respectively.

The key idea of 1.58-bit training is to maintain 16-bit precision “shadow weights” and quantize them on-the-fly. We follow the basic formulation of BitNet (Wang et al., 2023) for replacing activations and weights in all linear layers of a language model as follows:

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

where $\mathbf{W}$ denotes the weight parameters of the linear layers ("shadow weights") and $\hat{\mathbf{I}}$ denotes the normalized input. Q_b defines the integer range of the activations, which defaults to 8 bits, i.e., 256 possible values. The scaling factors w_{scale} and x_{scale} are derived from the respective means of the weight matrix $\mathbf{W}$ and the layer's input $\hat{\mathbf{I}}$.

3.2 Continual 1.58-bit Pre-training

We hypothesize that, to obtain the best possible 1.58-bit model given a fixed amount of data, the model needs to first find a good set of 16-bit parameters before those can be quantized to 1.58-bit. As such, we hypothesize that there exists a point $t^\star$ during training, at which one can switch from 16-bit training to 1.58-bit training in order to achieve an ultimately better 1.58-bit model, than one would obtain by complete quantization-aware training on the same data.

Formally, given a training set $\{\mathbf{x_i}\}_{i<N}$, we hypothesize that there exists a specific point in training $t^\star$, such that a model first trained with 16 bit on $\mathbf{x_0}, \mathbf{x_1}, \ldots, \mathbf{x_{t^\star}}$ and then trained with 1.58 bit on $\mathbf{x_{t^\star+1}}, \mathbf{x_{t^\star+2}} \ldots, \mathbf{x_{N-1}}$ ultimately performs better than a model pre-trained with 1.58-bits on the full training set $\mathbf{x_0}, \mathbf{x_1} \ldots \mathbf{x_{N-1}}$. See Figure 1. At the transition point $t^\star$, the 16-bit weights turn into "shadow weights" for quantization-aware training.

3.3 Optimizer State Retention

A key consideration in investigating continual 1.58-bit pre-training is the availability of optimizer states. If the optimizer states are available, we expect a smooth transition from pre-training to continued pre-training. In our strictly controlled experimental setup, the optimizer states are available. However, in practice, optimizer states cannot be assumed to be available, when one would seek continuing pre-training on arbitrary base models. We take the opportunity to investigate the effect of optimizer states being available vs. assuming that they were not available.

3.4 Phasing in Quantization Strength

We further investigate a recently proposed technique to gradually increase quantization strength. In this formulation, we would train a model on $\mathbf{x_0}, \mathbf{x_1}, \ldots, \mathbf{x_s}$ with 16-bit training, and then gradually introduce quantization strength on the data $\mathbf{x_{s+1}}, \mathbf{x_{s+2}}, \ldots \mathbf{x_{t^\star}}$ (Mekkouri et al., 2024). Specifically, we consider an extra hyperparameter λ integrated into the calculation of $\mathbf{x}_{\text{quant}}$ and $\mathbf{w}_{\text{quant}}$ denoting the activations and weight quantization respectively. We formulate the gradual quantization strength as:

$$\mathbf{x}_{\text{softquant}} = \mathbf{x} + \lambda \cdot \text{detach}\left(\mathbf{x}_{\text{quant}} - \mathbf{x}\right)$$
$$\mathbf{W}_{\text{softquant}} = \mathbf{W} + \lambda \cdot \text{detach}\left(\mathbf{W}_{\text{quant}} - \mathbf{W}\right)$$

where λ is a hyperparameter that controls the quantization strength, and $\text{detach}(\cdot)$ prevents gradient flow by detaching the operations from the computation graph. Reflecting on previous work (Mekkouri et al., 2024) and experimenting with different schedules, we observed that the main effect of using a gradual phasing in of quantization strength seems to happen in the transition from near-full quantization to full quantization, i.e., for high values of λ. In our experiments, the value for the hyperparameter λ at each optimization step is therefore determined by the following schedule:

$$\lambda(t) = \begin{cases} 0, & \text{if } t \leq s \\ 2\sigma(\frac{5(t-s)}{t^\star - s}) - 1, & \text{if } s < t \leq t^\star \\ 1, & \text{otherwise} \end{cases}$$

Here, σ is the logistic sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

This schedule is based on the shifted and scaled part of the sigmoid function for non-negative x.

4 Experimental Setup

We conducted experiments investigating the potential of continuing the pretraining in 1.58-bits from (partially) pre-trained 16-bit models, initializing the 1.58-bit model's "shadow-weights" with the pre-trained 16-bit weights. Furthermore, we investigate the impact of optimizer state retention and phasing in quantization strength.

Architecture Specifically, we employ the model architecture of Open Language Models (Groeneveld et al., 2024) in their official 1B parameters configuration. We use the BitLinear package[1] to all replace all `nn.Linear` layers within an

[1]https://pypi.org/project/bitlinear/

OLMo model by `BitLinear` layers (as described in Section 3), which includes both projection layers within the attention and feed-forward modules.

Training We train all models on the Dolma dataset (Soldaini et al., 2024), with OLMo's standard hyperparameters[2]. In particular, optimization is carried out by the AdamW optimizer (Kingma and Ba, 2015) with a cosine learning rate scheduler with warmup. We employ the same learning rates for both the 16 and 1.58-bit baselines in accordance with observations in our prior work (Nielsen and Schneider-Kamp, 2024; Nielsen et al., 2024). We employ a sequence length of 2048 and a batch size of 2048, totaling to a batch size of 4M tokens. Each experiment is run for a total of 10,000 optimizer steps.

Experimental Conditions In this setting, we compare various conditions:

- We vary the point in training, where we transition from 16 bit to 1.58 bit precision: either at 2K, 4K, or 6K steps.
- We control the retention of optimizer states: keeping the optimizer state alive vs. resetting the optimizer.
- We compare a sharp transition from 16-bit training to 1.58-bit training against a soft transition by gradually phasing-in quantization strength.

Baselines As our baselines, we consider training the model entirely under 1.58-bit quantization-aware training (for brevity: full 1.58-bit training) and entirely under standard 16-bit training (full 16-bit training). The expectation is that the performance of continually pre-trained 1.58-bit models will land between those two baselines, with full 16-bit training being expected to perform best.

Evaluation For downstream evaluation, we employ ARC-easy (reasoning Clark et al., 2018), CommitmentBank (De Marneffe et al., 2019), COPA (Roemmele et al., 2011), HellaSwag (Zellers et al., 2019), MRPC (Dolan and Brockett, 2005), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), RTE (Dagan et al., 2005), SciQ (Johannes Welbl, 2017), SST-2 (Socher et al., 2013), and WinoGrande (Sakaguchi et al., 2021). All downstream task evaluations are carried out via zero-shot inference, i.e., without fine-tuning and relying solely on pre-trained knowledge and generalization capabilities.

5 Results

We first present the results regarding 16-to-1.58-bit quantization-aware training in Section 5.1, followed by results comparing effects of and optimizer state retention (Section 5.2) and of phasing in of quantization strength (Section 5.3). Lastly, we report the results of the evaluation on downstream tasks in Section 5.4.

5.1 Results for Continual 1.58-bit Pre-training

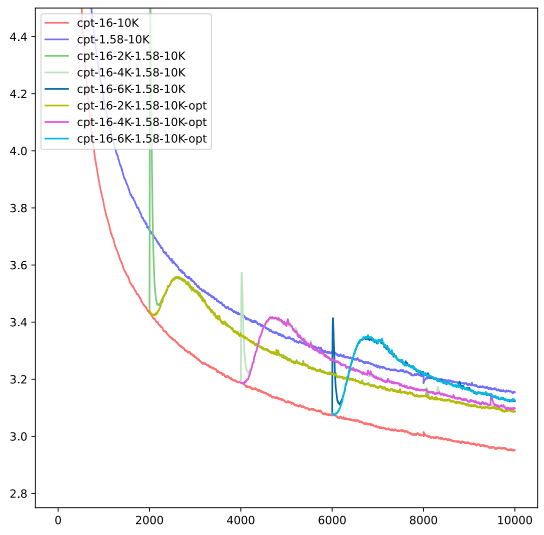
We conduct experiments investigating the potential in continuing the pre-training in 1.58-bits from a 16-bit models, initializing the 1.58-bit model's "shadow weights" with the pre-trained 16-bit weights. Further, we investigate the impact of gradually phasing in the quantization strength, and, lastly, continuing the 1.58-bit pre-training with the 16-bit pre-training optimizer.

We show that the best regime encountered in these pre-trainings consists of 2K 16-bit steps, demonstrating a gradually decrease in performance for both 4K and 6K 16-bit steps starting-points. Even more importantly, all three continual 1.58-bit training runs achieve a better performance than the full 1.58-bit training, demonstrating that training 1.58-bit models from scratch is suboptimal.

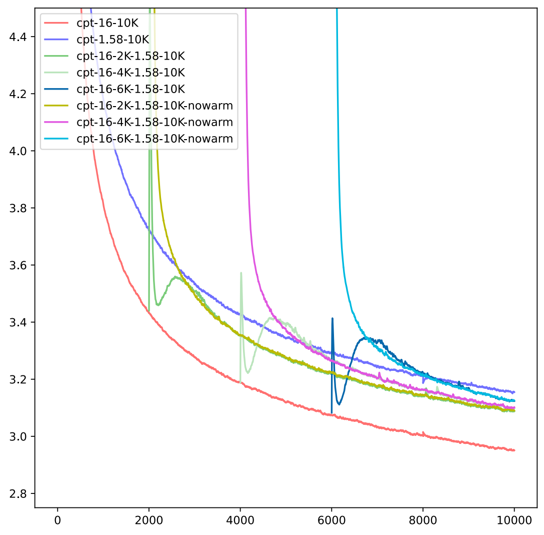
In Figure 2a, we observe that training a 16-bit model from scratch for 10K steps yields a loss value at 2.95 (red), whereas a fully 1.58-bit training achieves a loss around 3.15 (purple) after equally many steps. Continuing the pre-training from 2K 16-bit steps yields the best of the continued pre-training variants, with a resulting loss value around 3.088 (green). We observe that transitioning early yields a spike in the loss for around 100 steps before the model is able to catch up. Continuing from 4K 16-bit steps (light green) exhibits a smaller curve-spike, yielding a loss value at 3.097 after 10K steps. Last, continuing from 6K 16-bit steps (dark blue) exhibits an even smaller curve-spike yielding a loss value of around 3.12.

Overall, it is clear that training 1.58-bit from scratch is suboptimal. Furthermore, the training loss curves indicate that 16-bit training for more steps is likely detrimental to the overall training loss. The choice of 2K 16-bit steps seems to be

[2]https://github.com/allenai/OLMo/blob/main/configs/official-0724/OLMo-1B.yaml



(a) Comparing the 16-bit training with different stages of 1.58-bit continued pretraining and investigating the impact of transferring the optimizer-states. Training runs marked with the "`-opt`" suffix have their optimizer state retained.



(b) Comparing the 16-bit training with different stages of 1.58-bit continued pretraining and investigating the impact of phasing in quantization. Training runs marked with the "`-nowarm`" suffix do not employ phasing in.

Figure 2: Training loss curves comparing the effect of different variants of 1.58-bit continual pre-training of 1B OLMo models from 16-bit into 1.58-bit models. Our baseline is full 1.58-bit quantization-aware pre-training (cpt-1.58-10K; blue). Standard 16-bit training without quantization (cpt-16-10K) is displayed in red. Continual pre-trainings, i.e., transitioning from 16-bit into 1.58-bit training, are marked according to the transition points at 2K, 4K, and 6K optimizer steps, respectively. All models have been trained for 10K steps in total. All training loss curves have been smoothed with an exponential filter with window size 64.

optimal within the four regimes considered.

5.2 Effect of Optimizer State Retention

We investigate the impact of retaining the optimizer from the 16-bit pre-training, a warm optimizer, instead of constructing a newly initialized optimizer. We observe that spikes in loss curves in Figure 2a can be dampened by employing the optimizer states from the 16-bit training, demonstrated by the experiments with yellow, pink, and teal loss curves, yielding a lower and smoother spike-curve across all transfers. However, we observe the smoothening in the transition is gradually having a smaller effect, when transitioning at later stages (4K and 6K steps) of the training. More importantly, the warm-optimizer runs resulted in comparable loss values to their corresponding cold-optimizer runs.

5.3 Effect of Phasing in Quantization Strength

Figure 2b shows the impact of phasing in quantization strength. In the "`nowarm`" case, the training loss exhibits large spikes, which are however recovered relatively soon. After 10K total steps, the training loss is again at the same level compared to gradually phasing in quantization strength. This suggests that phasing in quantization strength is not having a lasting impact when pre-training language models, and that there is sufficient time to recover from any disturbance imposed by the abrupt quantization. It seems that quantization may as well be introduced at one point in time, alleviating the need for tuning extra hyperparameters and schedules determining the quantization strength.

5.4 Results of the Downstream Evaluation

We evaluated the downstream performance on all downstream tasks after each 1K optimizer steps.

We report the results of the final downstream evaluation after 10K steps in Table 1. Figure 3 shows the trajectories of downstream task performance over the course of the training runs. Notably, full 16-bit training only achieves the best result on the HellaSwag dataset. Full 1.58-bit training achieves the best result on the SciQ dataset. On all other datasets, one of the continually pre-trained models transitioning into 1.58-bit training at 2K, 4K or 6K steps attains the best downstream evaluation result. That said, the margin between the downstream results are small and, as expected, the performance of the 16-bit model is on average higher than any specific configuration of the 1.58-bit models. Given the relative volatility of some

Model	ARC-easy	CommitB	COPA	HellaSwag	MRPC	OpenBookQA	PIQA	RTE	SciQ	SST-2	WinoGrande
full 1.58-bit training	0.4561	0.4107	0.6300	0.3212	*0.8122*	*0.2760*	0.6425	0.5343	**0.6840**	*0.5803*	*0.5185*
full 16-bit training	*0.4596*	0.4107	*0.6500*	**0.3607**	*0.8122*	*0.2760*	*0.6589*	0.5379	0.6780	0.5447	0.5170
cpt-16-2K-1.58-10K	0.4526	*0.4464*	**0.6900**	0.3375	0.8105	0.2700	**0.6621**	0.4874	0.6760	0.5539	0.5146
cpt-16-2K-1.58-10K-nowarm	0.4456	0.4107	0.6300	0.3342	**0.8134**	0.2700	0.6480	*0.5487*	0.6760	**0.6101**	0.4988
cpt-16-2K-1.58-10K-opt	**0.4632**	**0.5179**	0.6300	*0.3376*	0.7951	0.2620	0.6420	0.4946	*0.6830*	0.4989	**0.5264**
cpt-16-4K-1.58-10K	0.4544	0.4286	*0.6500*	0.3328	*0.8122*	0.2680	0.6458	0.5235	0.6520	0.5229	0.5146
cpt-16-4K-1.58-10K-nowarm	0.4491	0.4286	0.6300	0.3329	*0.8122*	0.2560	0.6415	**0.5668**	0.6660	0.5745	0.5107
cpt-16-4K-1.58-10K-opt	0.4439	0.4107	0.6200	0.3339	*0.8122*	**0.2780**	0.6491	0.5451	0.6600	0.5092	0.4996
cpt-16-6K-1.58-10K	0.4351	0.3929	0.6000	0.3262	*0.8122*	0.2560	0.6366	0.5451	0.6790	0.5791	0.4838
cpt-16-6K-1.58-10K-nowarm	0.4491	0.4286	0.6100	0.3240	*0.8122*	0.2640	0.6398	*0.5487*	0.6700	0.5183	0.4988
cpt-16-6K-1.58-10K-opt	0.4333	0.3750	0.6100	0.3291	*0.8122*	*0.2760*	0.6360	0.5090	0.6810	0.5493	0.5107

Table 1: Final downstream evaluation with the best results marked in **bold** and the runner-ups in *italics*. All model variants have been trained for 10K steps in total.

of the downstream evaluation results and the small differences, it remains unclear to what degree these differences are statistically significant.

6 Discussion

Through a comprehensive set of experiments where we have full control over the pre-training data and regimen, we have shown that, counterintuitively, quantization-aware training of 1.58-bit models from scratch is neither ideal nor most efficient for obtaining the best possible 1.58-bit models. In addition, we find that previously proposed strategies of gradually phasing in quantization strength are not needed as long as there are sufficient optimization steps to catch up. Furthermore, we make similar observations for the optimizer states: Resetting the optimizer results in a spike in training loss, but the loss can be recovered after relatively few further optimization steps. Crucially, our experiments on downstream tasks indicate that continual 16-to-1.58-bit pre-training is a highly effective training strategy, consistently outperforming full 1.58-bit training, and sometimes even exceeding downstream performance of 16-bit models.

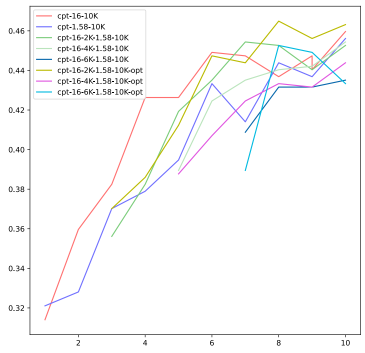
Our results show that a period of standard training at higher precision (16-bit), and then continuing the pre-training with quantization-aware 1.58-bit training constitutes a preferable training regimen.

We hypothesize that this is because of particular challenges that 1.58-bit quantization-aware training imposes on optimization. Adjusting the weights is more difficult under 1.58-bit training, because the quantized weight may very well remain stable even though the "shadow weights" change and their gradient is estimated straight-through. Thus, starting from random parameters, as opposed to a set of parameters obtained through 16-bit training, seems to be more challenging to optimize. Interestingly, the continual pre-training variants with an initial period of standard training have led to lower overall training loss compared to 1.58-bit training from the start, even when being restricted to the same amount of data and number of steps.

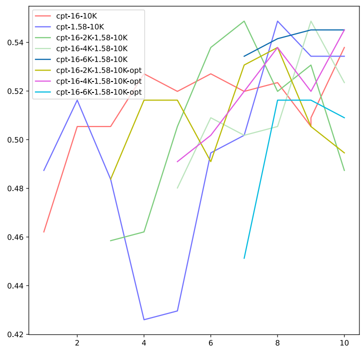
Re-initializing the optimizer when transitioning to 1.58 bit training yields spikes in the loss-curves caused by the loss of the momentum terms – both with and without quantization phase-in. Retaining the optimizer states or phasing in quantization strength dampens this effect substantially. Notably, without phased-in quantization strength, the spikes have a much higher magnitude. However, these effects are dampened after a few optimization steps and the tweaks to mitigate the spikes do not yield a gain in final performance. We hypothesize that when retaining the optimizer state, the optimizer still needs to react to the coarseness of the 1.58 bit representations as well as the new type of feedback though the straight-through estimator, giving a similar yet shorter lasting spikes. In the end, if there is sufficient data available for a moderate amount of optimization steps, then it is not necessary to avoid the loss spikes.

This finding has important implications: If one aims for the best possible, most efficiently trained, 1.58-bit model given a fixed amount of data, one should still first train a 16-bit model on a proportion of the data (e.g., between 20 and 40%) and only then introduce quantization into the training. Keeping the optimizer state is beneficial to avoid loss spikes, but if the optimizer state is not available, sufficient further optimization steps enable the model to recover, eliminating the need for optimizer checkpoints – which is crucial for continual 1.58-bit pre-training of existing models. Alleviating inference compute demands comes with immediate gains for democratization of AI research (Xiao et al., 2023) and for advances in scaling test-time compute.

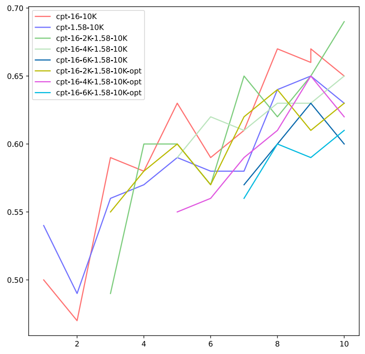
From a different perspective, our results further support the idea of 1.58 continual pre-training be-



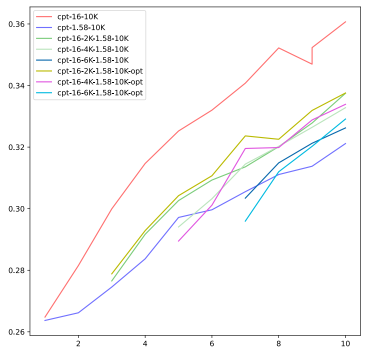
(a) ARC-Easy (Clark et al., 2018)



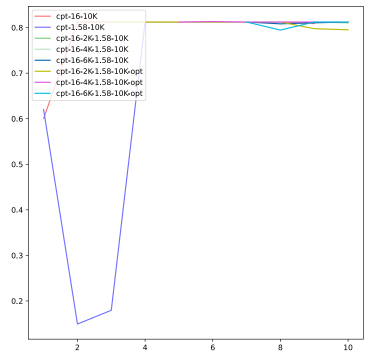
(b) RTE (Dagan et al., 2005)



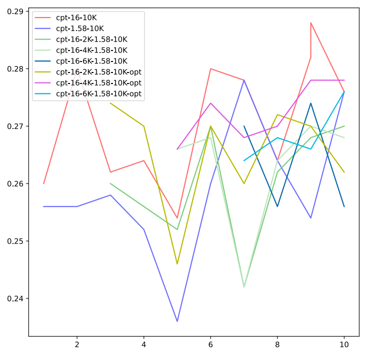
(c) COPA (Roemmele et al., 2011)



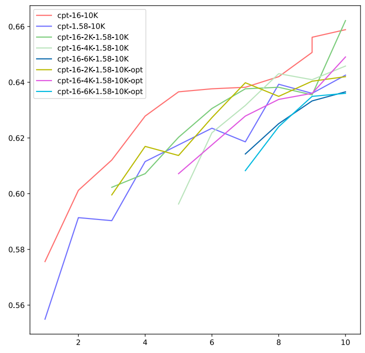
(d) HellaSwag (Zellers et al., 2019)



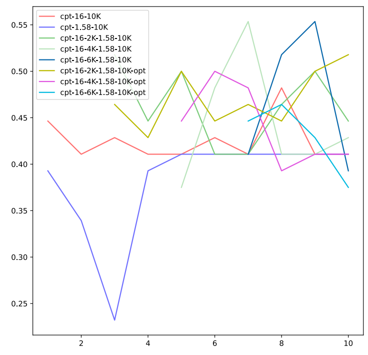
(e) MRPC (Dolan and Brockett, 2005)



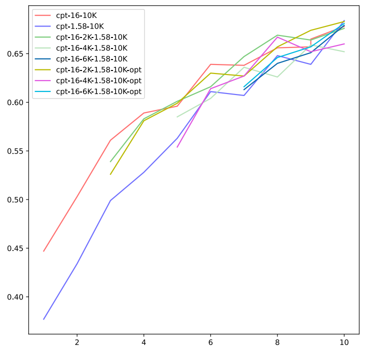
(f) OpenBookQA (Mihaylov et al., 2018)



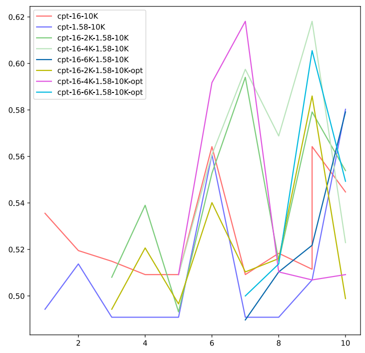
(g) PIQA (Bisk et al., 2020)



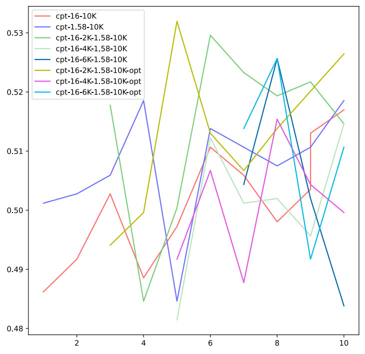
(h) CommitmentBank (De Marneffe et al., 2019)



(i) SciQ (Johannes Welbl, 2017)



(j) SST-2(Socher et al., 2013)



(k) WinoGrande (Sakaguchi et al., 2021)

Figure 3: Downstream evaluation of full 16-bit, continually pre-trained 1.58-bit, and full 1.58 trainings. The y axes are not scaled such that relative differences are more visible.

ing applied to fully pre-trained base models – opening avenues of research to convert arbitrary 16-bit pre-trained models into corresponding 1.58-bit models. Determining the minimum amount of data required for this conversion is an interesting direction for future work. Unlike post-quantization methods, quantization-aware training schemes enable lossless quantization (considering the difference training and inference performance). This is particularly important in safety-critical domains (e.g., medical), where "blind" post-training quantization could lead to an unexpected degradation of performance. The deployment of 1.58 models requires less memory, due to the fact that 5 ternary weights and be packed into one 8-bit integer (Couture-Harpin, 2024). Naturally, this will also speed up the hardware load-times with a small overhead of unpacking the weights online, during inference. The computational load is also reduced by the replacement of multiplication and subsequent addition with addition alone. However, fully realizing these benefits in practice requires more specialized hardware implementation, which is already an activate area (Microsoft, 2024). Future hardware generations designed to include support for this scheme will further enhance efficiency

We further demonstrate that continually pre-trained 1.58-bit models are competitive on a broad range of downstream tasks, showcasing the capabilities of these inference-efficient models. Notably, the number of parameters was kept fixed between conditions in all our experiments, indicating that those continual 1.58-bit pre-training goes beyond recently proposed scaling laws for quantized networks in general (Kumar et al., 2025) and BitNet-style quantization-aware training (Nielsen and Schneider-Kamp, 2024) in particular. However, an analysis of the scaling behavior of our proposed continual quantization-aware pre-training strategy is left for future work. A further promising direction for future work would be to study the effect of 1.58-bit instruction finetuning, e.g., whether instruction tuning data might be sufficient for converting existing base models from 16 to 1.58 bits.

7 Conclusion

We conducted a systematic comparison between language models trained from scratch with 1.58-bit precision and 16-to-1.58-bit continual pre-training paradigms. Our results show the existence of a data-optimal transition point for switching from 16-bit to 1.58-bit training, providing insights into efficient low-bit training strategies. We evaluated the downstream performance of 1.58-bit models, highlighting their viability for real-world applications. These findings contribute to the broader discussion on low-bit training efficiency and its implications for scalable AI model development. Future work may determine the minimal amount of data needed to successfully convert a 16-bit model into a 1.58-bit model through continual pre-training.

8 Limitations

The performance on downstream tasks are expected to be further increased when the models undergo supervised instruction fine-tuning and preference optimization before evaluation. We expect that all models benefit similarly from instruction fine-tuning. However, at this point in time, it cannot be excluded that 16-bit models benefit more from instruction fine-tuning and preference optimizaiton than 1.58-bit models. Future work may specifically investigate the effects of 1.58-bit quantization-aware training during instruction fine-tuning and preference alignment.

9 Ethical Considerations

Our work aims at improving the inference compute demands of langauge models. It may contribute to the democratization of AI, alleviating privacy concerns, and to reduce the environmental footprint of LLMs. In particular, our findings suggest that large language models may be converted into lower bit-precision through continual pre-training. We acknowledge that this may come with ethical challenges that govern all research on making powerful models more accessible (Bengio et al., 2025). However, this work is purely scientific and does not promote easier access to an actual pool of very powerful models. Using our proposed training strategy would require a similar compute budget as it is needed for standard pre-training.

Acknowledgements

We acknowledge EuroHPC JU for awarding the project ID EHPC-AI-2024A04-086 access to Leonardo Booster. This research was further in parts supported by the Danish Foundation Models (DFM) project.

References

Malyaban Bal, Yi Jiang, and Abhronil Sengupta. 2024. Exploring extreme quantization in spiking language models. *Preprint*, arXiv:2405.02543.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *Preprint*, arXiv:1308.3432.

Yoshua Bengio, Sören Mindermann, Daniel Privitera, Tamay Besiroglu, Rishi Bommasani, Stephen Casper, Yejin Choi, Philip Fox, Ben Garfinkel, Danielle Goldfarb, et al. 2025. International AI safety report. *Preprint*, arXiv:2501.17805.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about physical commonsense in natural language. In *AAAI*.

Peng Chen, Bohan Zhuang, and Chunhua Shen. 2021. Fatnn: Fast and accurate ternary neural networks. In *ICCV*, pages 5219–5228.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *Preprint*, arXiv:1803.05457.

Francis Couture-Harpin. 2024. How to pack ternary numbers in 8-bit bytes.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *Preprint*, arXiv:2210.17323.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, A. Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Daniel Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hanna Hajishirzi. 2024. Olmo: Accelerating the science of language models. *Preprint*, arXiv:2402.00838.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Preprint*, arXiv:2501.12948.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. OpenAI o1 system card. *Preprint*, arXiv:2412.16720.

Matt Gardner Johannes Welbl, Nelson F. Liu. 2017. Crowdsourcing multiple choice science questions. *Preprint*, arXiv:1707.06209v1.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Tanishq Kumar, Zachary Ankner, Benjamin Frederick Spector, Blake Bordelon, Niklas Muennighoff, Mansheej Paul, Cengiz Pehlevan, Christopher Re, and Aditi Raghunathan. 2025. Scaling laws for precision. In *ICLR*.

Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. 2016. Ternary weight networks. *Preprint*, arXiv:1605.04711.

Zhikai Li and Qingyi Gu. 2023. I-ViT: Integer-only quantization for efficient vision transformer inference. In *ICCV*, pages 17065–17075.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for llm compression and acceleration. *Preprint*, arXiv:2306.00978.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023. LLM-QAT: Data-free quantization aware training for large language models. *Preprint*, arXiv:2305.17888.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. 2024. The era of 1-bit llms: All large language models are in 1.58 bits. *Preprint*, arXiv:2402.17764.

Mohamed Mekkouri, Marc Sun, Leandro von Werra, Predo Cuenca, Omar Sanseviero, and Thomas Wolf. 2024. Fine-tuning LLMs to 1.58bit: extreme quantization made easy — huggingface.co. `https://huggingface.co/blog/1_58_llm_extreme_quantization`. [Accessed 04-02-2025].

Microsoft. 2024. Official inference framework for 1-bit llms.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

Jacob Nielsen, Lukas Galke, and Peter Schneider-Kamp. 2024. When are 1.58 bits enough? a bottom-up exploration of bitnet quantization. *Preprint*, arXiv:2411.05882.

Jacob Nielsen and Peter Schneider-Kamp. 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.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI spring symposium series*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020. Green AI. *Commun. ACM*, 63(12):54–63.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642. ACL.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. 2024. Dolma: An Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. *arXiv preprint arXiv:2402.00159*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2020. Energy and policy considerations for modern deep learning research. In *AAAI*, pages 13693–13696. AAAI Press.

Jainaveen Sundaram and Ravishankar Iyer. 2024. LLaVaOLMoBitnet1B: Ternary LLM goes multimodal! *Preprint*, arXiv:2408.13402.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. 2023. BitNet: Scaling 1-bit transformers for large language models. *Preprint*, arXiV:2310.11453.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *ICML*, volume 202, pages 38087–38099. PMLR.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. 2023. QA-LoRA: Quantization-aware low-rank adaptation of large language models. *Preprint*, arXiv:2309.14717.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? *Preprint*, arXiv:1905.07830.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. 2016. Trained ternary quantization. *Preprint*, arXiv:1612.01064.