[1] Abhi Akshat
[2] Kritika Tripathi
[3] Devanshi Malik
[4] Kusum Lata

1 Bit LLM for Hindi Language at Lower Memory Consumption

Journal of Electrical Systems

Abstract: - With the ever-increasing size of Large Language models, an approach to reduce the memory consumption has become vital. Techniques like parameter scaling and instruction tuning are commonly used to improve LLM performance. Although effective, these methods greatly increase memory consumption and computational complexity, requiring more significant resources for model training and inference. Model compression methods like quantisation are becoming more popular as a solution to these problems. One such method is Bitnet, specifically Bitnet-1.58, here we quantize the weights in such a way that it takes one of the tree values, 0,1 or -1. Since there is little research on quantization of Hindi LLMs, this paper explores the Bitnet-1.58 architecture for Hindi LLM based on the LLaMA architecture. A 100 million and a 1 Billion parameter model was trained on a 2 billion token dataset. A perplexity of almost 42 and 12 was achieved respectively. Although the performance was not at par with the current state of the art model, but given the limited resources, the performance achieved was good. The model remembered the training dataset sufficiently well, although the performance in general NLP tasks was suboptimal. Given enough computational resources and large datasets, the models could perform at par with full precision models. Training was conducted on two NVIDIA A100 GPUs.

Keywords: Large Language Models, LLMs, NLP, Model Compression, Model Quantization, Bitnet, Hindi LLMs.

I. INTRODUCTION

In recent times, Large Language Models (LLMs) [1] have become one of the most well-liked and advanced artificial intelligence systems. The Transformer design used by LLMs is particularly good at handling sequential data, such as text. When producing predictions, the Transformer model is able to concentrate on particular portions of the input because it is dependent on mechanisms such as attention. They can be optimized for tasks particular to a certain domain and trained on a variety of datasets. There are multiple LLMs, like Large Language Model Meta AI (LLaMA) [2], Generative Pre-trained Transformers (GPT) [3], Falcon [4], [5], BLOOM [5] and Mistral [6] that are performing extremely well.

These models perform exceptionally well because of the volume of data they are trained on and the computations that went into their development. This is where the issue of computational demands and model sizes arises. With 405 billion parameters, Llama 3.1 405B [7] needs about 800 GB of RAM to be served in its original FP16 precision. Even with half-precision, it cannot fit on a single node with 8 X H100 80GB GPUs; therefore, it must be deployed on many nodes, on lower precision, or on both. As a result, in order to address this problem, different model compression strategies are being investigated in order to lower these high parameter models' memory needs. Model compression is an emerging technique as it helps in reducing the resource requirements of a model and also increase the inference speed. Usually, the memory requirement of the models is directly proportional to the number of parameters in the model. **Table 1** explore some of the most popular models with their number of parameters and model size.

Model compression [8] is made possible by methods including quantization, parameter sharing, pruning, and knowledge distillation.

[1, 2] Department of Computer Science and Engineering, Sharda University, Greater Noida, Uttar Pradesh, India, 201310.
Email: abhiakshatofficial@gmail.com
[2] Department of Computer Science and Engineering, Sharda University, Greater Noida, Uttar Pradesh, India, 201310.
Email: kritika.tripathi03@gmail.com
[3] Department of Computer Science and Engineering, Sharda University, Greater Noida, Uttar Pradesh, India, 201310.
Email: devanshimalikd@gmail.com
[4] Department of Computer Science and Engineering, Sharda University, Greater Noida, Uttar Pradesh, India, 201310.
Email: kusumlata.1@sharda.ac.in

Table 1. Size of Various Models

Model Name	Size	Number Of Parameters
GPT – 4	1.5TB+ (Estimated)	1.8T (Estimated)
LLaMA 2	140 GB	70B
LLaMA 3.1	800GB	405B
Mixtral 8x7B	25GB	12B
Qwen-72B	144GB	72B
Falcon 108B	180GB	108B
Claude 2	274GB	137B

On the comparison of these techniques quantization was found to show one of the most significant results for reducing the memory requirements of such models. The origins of quantization [9] can be traced to the use of vector quantization to lower computation cost in voice and image neural networks. On the other hand, contemporary studies concentrate on quantization as a means of lowering computation and memory overhead.

Numerous novel quantization techniques are being developed in the field of LLMs with the goal of quantifying gradients, weights, activations, and even KV cache. Initially, 8-bit quantization of models was the goal of early LLM quantization techniques; later, this number decreased to 4, 3, 2, and even 1-bit model widths.

The Hindi language, among the languages that are spoken the most often worldwide, has a rich linguistic structure that presents unique challenges in language modeling. While much research has been conducted on large language models in English and other major languages, there has been comparatively less focus on building and optimizing models for Hindi. This gap in research highlights the need for efficient, scalable models tailored to Hindi, especially given the rapid growth in digital content and NLP applications in India and other Hindi-speaking regions. LLMs like Airavata-7B [10], Ganga-1B and some others have been developed, which are based on Hindi.

The bit allocations for the sign, exponent, and mantissa across several floating-point (FP) and integer (INT) formats are important and display the number representation formats that are typically used for LLMs. With 23 bits for the mantissa, FP32 (32-bit floating point) offers the highest precision but uses more memory. While FP8 (8-bit) further compresses data with only 2 bits for the mantissa, sacrificing some precision, FP16 (16-bit) uses 10 bits for the mantissa, reducing memory use. In contrast, integer formats such as INT8 and INT4 are less accurate but very effective for model compression since they store numbers without an exponent or mantissa. In quantisation, lower bit-width formats (FP8, INT8, INT4) are frequently employed to increase memory efficiency and speed up inference at the modest expense of accuracy. Integer representations optimise computational efficiency, while floating-point forms provide improved numerical stability. These adjustments are essential for increasing the deployment viability of large-scale AI models.

This paper explores the Bitnet-1.58 [11] architecture for Hindi language. Bitnet-1.58 is a architecture that based on LLaMA and uses a ternary weight system, that is, the weights or parameters take on one the following values, 1,0 or -1, hence 1.58 bits. Comparing with the full precision model the reduction is size is immense. Let's say a model with 7B parameters is stored in 32 bits, it consumes roughly 28GB of memory, where a model stored in 1.58 bits will consume 800MB. The immense reduction in the size of model provides the motivation and excitement to explore this architecture.

II. Literature Survey

Wang et al. [11] presented 1-bit LLM variant, BitNet b1.58 based on BitNet architecture, in which every single parameter of the LLM is ternary {-1, 0, 1}. It establishes a novel scaling law and method for developing new generations of LLMs that are both high-performing and cost-effective. They provided a Pareto solution to reduce latency, throughput, and energy while maintaining model performance of LLMs and specifically for quantization function Round Clip method is being used. This model with a size of 3.9B parameters use 2.38(3.32x) memory (GB) with a latency of 2.11(2.40x) and gives a perplexity score of 9.62. The BitNet b1.58's throughput with 70B parameters is 2977(8.9x) where as throughput of LLaMA LLM with 70B parameters is 333(1.0x).

Dong et al. [12] introduced STBLLM, a structural binarization model for compressing LLM to sub-one-bit precision without fine-tuning. It is on par with BiLLM and has a better bit-width-perplexity trade-off for 7B to 65B models.

Han et al. [13] introduced a 1-bit model compression framework namely OneBit, which contain a 1-bit parameter representation approach to better quantise LLMs, an effective parameter initialisation method based on matrix decomposition to increase the convergence speed of the quantisation framework, and quantised the weight matrix using the function Sign(.). OneBit method achieves a perplexity score of 9.18 in the Wiki2 dataset on LLaMA-13B model and the FP16 baseline is 5.09.

Wang et al. [14] introduced BitNet, a scalable and robust 1-bit Transformer architecture suitable for large language models. They introduce BitLinear as a drop-in replacement of the nn.Linear layer in order to train 1-bit weights from scratch. BitNet has a scaling law similar to full-precision Transformers, indicating that it may effectively scale to even enormous language models while preserving efficiency and performance. The capabilities' scaling efficiency is substantially higher than the FP16 Transformer baseline, in terms of both zero-shot and few-shot performance.

Song et al. [15] Introducing bitnet.cpp, a specialised software stack to unleash the full power of 1-bit LLMs, and a set of kernels must be constructed to facilitate rapid and lossless inference of ternary BitNet b1.58 LLMs on CPUs. This model is evaluated in terms of inference speed and energy cost and use a 700M BitNet b1.58 model. BitNet provides considerable speedups, ranging from 2.37x to 6.17x on x86 CPUs and 1.37x to 5.07x on ARM CPUs.

Wang et al. [16] proposed a promising direction for reducing the inference cost of LLMs while maintaining their performance. They introduced BitNet a4.8 which enables 4-bit activations for 1-bit LLMs by employing a hybrid quantization and sparsification strategy to mitigate the quantization errors introduce by the outlier channels. They also demonstrate that BitNet a4.8 achieves performance comparable to BitNet b1.58 with equivalent training cost. BitNet a4.8 achieves an overall sparsity of 44.5% whereas BitNet b1.58 achieves an overall sparsity of 7.3%.

Zhou et al. [17] proposed Bitnet.cpp, an optimized inference system for BitNet b1.58 and ternary LLMs. It incorporates a novel mixed-precision matrix multiplication (mpGEMM) library to facilitate sub-2-bits-per-weight, efficient and lossless inference. The library features two core solutions: Ternary Lookup Table (TL) and Int2 with a scale (I2_S). They demonstrated that Bitnet.cpp can boost performance by up to 6.25x over full-precision baselines and up to 2.32x over low-bit baselines.

Chee et al. [18] introduced QuIP, a PTQ method using incoherent weight and Hessian matrices for effective 2-bit quantization. QuIP attains near full-precision accuracy at high throughput for big LLMs such as OPT-66B with adaptive rounding and computational efficiency.

Xu et al. [19] presented techniques such as intra-weight mixed-precision, 2-bit sparse outliers, and asynchronous dequantization for fast LLM inference on GPUs. These techniques provide up to 2.53x speedup and significant cost savings on models such as LLaMA2.

Zhang et al. [20] introduced VPTQ, an efficient vector post-training quantization technique with Second-Order Optimization and Effective Codebook Initialization. VPTQ lowers perplexity dramatically on LLaMA-2 and LLaMA-3 models at 2-bit precision compared to prior state-of-the-art techniques.

Liu et al. [21] presented ParetoQ, a shared framework for 1 to 4-bit comparison, and illustrated a dramatic learning transition under 3 bits. Their 600M ternary model is more efficient than past 3B models and contains only one-fifth the number of parameters.

Zhao et al. [22] proposed Atom, a low-bit quantization method that speeds up serving and reduces memory use with low-bit operators. Atom achieves up to 7.73x throughputs improvement relative to FP16 while maintaining high accuracy with fine-grained quantization and mixed-precision.

Chen et al. [23] proposed BitMoD, an algorithm-hardware co-design that accelerates LLMs with fine-grained data type adaptation to 3-bit quantization. BitMoD achieves up to 1.69x and 1.48x speedups over ANT and OliVe, respectively, compared to state-of-the-art accelerators.

Huang et al. [24] introduced BiLLM, a post-training 1-bit quantization method with optimal splitting search for efficient binarization of LLMs. BiLLM provides high accuracy (8.41 perplexity on LLaMA2-70B) with fast processing in a significant way compared to earlier quantization techniques.

Kim et al. [25] presented SqueezeLLM, a post-training quantization technique based on sensitivity-guided non-uniform quantization and dense-sparse decomposition. It achieves 2.3x speedup and 2.1x reduced perplexity gap at 3-bit precision in LLaMA models under the same memory constraint.

Li et al. [26] presented TesseraQ, a post-training quantization technique with progressive adaptive rounding for ultra-low-bit LLM weight quantization. It greatly enhances Wikitext2 perplexity and accelerates 2-bit weight-only downstream accuracy on LLaMA-2-7B.

Yuan et al. [27] proposed PB-LLM, a partially-binarized LLM model that preserves reasoning ability at ultra-low-bit quantization. Utilizing both PTQ and QAT, it applies salient weights and super-scaling to alleviate quantization error, achieving stable performance like 401.65 perplexity on LLaMA-7B (C4 dataset).

Lin et al. [28] introduced AWQ, an activation-aware weight quantization technique that holds 1% salient weights to mitigate the quantization error. It outperforms previous techniques and, combined with TinyChat, provides more than 3x speedup on 4-bit LLMs, and this makes LLaMA-2 70B efficient enough to deploy on-device on mobile GPUs.

Guo et al. [29] introduced PT-BitNet, a post-training quantization method based on ternary quantization on LLMs up to 70B parameters. With a two-stage algorithm, it reaches 61% average downstream accuracy, outperforming BitNet b.158's 51.2%.

Liu et al. [30] proposed LLM-FP4, a 4-bit floating-point quantization method for LLM weights and activations, which is better than integer quantization in variable data distributions and has an average score of 63.1 for zero-shot reasoning tasks on LLaMA-13B.

Zhang et al. [31] introduced BitDistiller, which integrates Quantization Aware Training and knowledge distillation for sub-4-bit LLMs. With a new CAKLD objective, it obtains better performance in 2-bit and 3-bit regimes at the cost of being more data and resource efficient.

Liu et al. [32] proposed LLM-QAT, data-free distillation of model-generated outputs with quantization-aware training. It enables sub-8-bit quantization, KV cache compression, and works well for all LLaMA models at up to 4 bits, enhancing throughput and long-sequence processing.

Xia et al. [33] proposed FP6-LLM, a 6-bit quantization technique with TC-FPx kernel design for fast LLM inference. It can perform LLaMA-70B inference on a single GPU and is 1.69x–2.65x faster than the FP16 baseline.

Chen et al. [34] introduced DB-LLM, a Dual-Binarization approach with Flexible Dual Binarization (FDB) and Deviation-Aware Distillation (DAD) for low-bit efficient LLMs. It improves representation efficiency, attains ~0.80 perplexity improvement, and lowers computational complexity with increased sparsity.

Raghavan et al. [35] introduced BitNet, an optimization technique that trains networks with integer-valued parameters by parameter range control via a differentiable upper bound. It allows for faster convergence, better model quality, and memory savings of large magnitudes.

III. Dataset Used

For the implementation of BitNet-1.58, a pre-training dataset was required. These datasets are invaluable resources for training and fine-tuning large language models (LLMs) for a wide range of natural language processing (NLP) tasks. Therefore, “odaigen_hindi_pre_trained_sp” dataset was used[36], which is a Hindi-based pre-training dataset with nearly 4 million rows containing long Hindi texts. This dataset combines a variety of datasets including, Wikipedia, Dialecthindi, Ai4bharat IndicParaphrase, Miracle Corpus and more summing up to around 21.9 million sentences. Each dataset offers a unique linguistic sense that can be used to develop comprehensive models of Indian language processing issues. This dataset was tokenized to 2 billion tokens. Researchers and developers can take advantage of this database as a rich source of building reliable, context-aware LLM models of specific language domains and applications.

IV. Methodology Used

This paper explores the Bitnet-1.58 for Hindi Language. Training and performance analysis of the model is conducted. **Figure 1** shows the steps that were followed for training and testing the model. The first step is to prepare data, this involves cleaning and tokenizing the data. For tokenizer tokenizer of AIRAVATA model [10] was used, it is one of the most accurate model for Hindi language. Using this tokenizer the dataset is converted to tokens and the result is about 2.7 billions tokens. After that, an architecture for model was created, in this case the BitNet-1.58 [11]was used, which is essentially a LLaMA architecture [2] which replaces the nn.linear layer with the BitLinear [14], more information about the architecture is mentioned later. After this, the dataset was passed to the model, calculate loss and update the weights. This is an overview of the entire training process. More details about training parameters are discussed later. After the training is completed testing was conducted using perplexity and IndicEval.

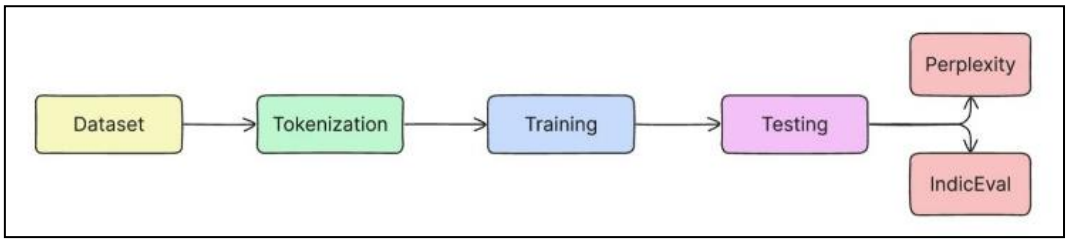

Figure 1. Flow of Research

A. BitNet b1.58

BitNet b1.58 [11], a quantisation method that represents the weights in the model using {-1, 0, 1}, is the foundation of this study. BitNet uses specialised layers called BitLinear that leverage ternary precision in place of conventional Linear layers in Multi-Head Attention and Feed-Forward Networks. By altering the conventional Linear (fully connected) layer to function effectively with low-bit quantisation, it preserves model performance while drastically cutting memory and computational expenses.

The remaining components remain high-precision, meaning they are 8-bits. Using the Absmean quantisation function, the weights are constrained to -1, 0, or +1. This function rounds each value to the closest integer between {-1, 0, +1} after first scaling the weight matrix by its average absolute value. The computation from the aforesaid method is shown in **equations (1), (2) and (3)**.

$$\widetilde{W} = RoundClip\left(\frac{W}{\gamma+\epsilon}, -1, 1\right), \tag{1}$$

$$RoundClip(x, a, b) = \max\left(a, \min\left(b, round(x)\right)\right), \tag{2}$$

$$\gamma = \frac{1}{mn}\sum_{ij}|W_{ij}| \tag{3}$$

Absmax quantisation, which scales activations to range $[-Q_b, Q_b]$ (Qb = 2b - 1) per token, is used to scale them to b-bit precision. In contrast to the BitNet implementation, BitNet b1.58 eliminates zero-point quantisation by scaling activations before the non-linear functions to the range $[0, Q_b]$. Both implementation and system-level optimisation benefit from this.

For stability and efficiency, the quantisation is done per token during inference and per tensor during training. After quantisation, the variance is maintained using LayerNorm. Model parallelism is used to divide matrix multiplication among several machines. Group quantisation, in which weights and activations are separated into groups and the parameters of each group are then independently estimated, is used to do this. This method aids in the scaling up LLMs.

The Straight Through Estimator method is employed by BitLinear. The STE allows weight updates using common gradient-based optimisation approaches by allowing the gradient to continue through as if the rounding had never taken place, rather than blocking it at the rounding step.

Because BitNet b1.58 employs SwiGLU, rotational embeddings, RMSNorm, and eliminates all biases, its design incorporates LlaMA-like components and is therefore practical enough to be incorporated into widely used open-source software. Compared to FP16 or FP32, the number of bits needed per parameter is significantly decreased because each weight only requires 1.58 bits.

By lowering memory consumption, increasing computational efficiency, and preserving accuracy, BitNet B1.58 performs better than 2-bit quantisation approaches. Its 1.58-bit quantisation scheme makes it possible to deploy transformer models at scale in a way that is more high-performance and hardware-friendly.

B. Training

Two language models were successfully trained: one with 100 million parameters and another with 1 billion parameters.

The training process was conducted on an NVIDIA A100-80GB GPU, the system had a RAM of 32GB and storage of 100GB. Leveraging the high memory bandwidth and computational efficiency to handle the large-scale operations required for model optimization.

Each model underwent two epochs of training, striking a balance between computational feasibility and model performance. The training process incorporated techniques such as gradient accumulation and mixed-precision training to optimize memory utilization and speed up convergence. Hyperparameters, including batch size, learning rate, and number of attention heads, were fine-tuned to ensure stable and effective training.

A comprehensive overview of the training configuration, including optimizer settings, learning rate schedule, and dataset composition, is provided in **Table 2**. Additionally, the training loss curve for the **1**-billion-parameter model, shown in **Figure 2** show the loss illustrates the progressive decline in loss, indicating the model's ability to learn and generalize effectively over training iterations.

Table 2. Model Training Parameters

Parameter	BitNet-1.58 100M	BitNet-1.58 1B
Batch Size	8	8
Context Size	256	256
Learning Rate	6×10^{-4}	3×10^{-4}
Number of Attention Heads	12	8
Dimension	768	2048
Layers	12	16
Intermediate Size	1024	4096

The final loss for 100M parameter model was 1.4 and for 1B model was 1.7. The loss achieved by the models was really good.

Despite successful training, certain challenges were encountered, such as memory constraints, high computational costs, and training stability issues. Future optimizations, including more extensive training cycles, better dataset pre-processing, and reinforcement learning techniques, could further enhance model performance.

C. Evaluation Metrics

Perplexity is employed for the model's assessment. The ability of language models to predict a sample of text is measured using a statistic called perplexity. It's calculated as the exponentiated average negative log probability assigned to each word by the model—lower perplexity indicates better predictive performance. The formula for perplexity is represented in **equation (4)**:

$$PPL\ (W) = \exp\left\{-\frac{1}{N}\sum_{i=1}^{N}\log \mathrm{p}\ (w_i)\right\} \quad (4)$$

Where W is the sequence of words, p (w_i) is each word's probability or likelihood as determined by the model w_i, and N is the sequence's overall word count.

The perplexity for the model was also calculated based on the training dataset used, therefore, odaigen_hindi_pre_trained_sp [36]. Roughly, 1% of this pretraining dataset was used to evaluate the performance of the models. Perplexity is employed for the model's assessment. The ability of language models to predict a sample of text is measured using a statistic called perplexity. It's calculated as the exponentiated average negative log probability assigned to each word by the model—lower perplexity indicates better predictive performance.

Apart from calculating the perplexity of models, IndicEval was used to evaluated the model's performance, which is a group of evaluation benchmarks used to evaluated Indic models, mainly for Hindi language. The model was evaluated on Indic NLU and Commonsense Reasoning tasks, which includes which evaluates natural language understanding (NLU) and commonsense reasoning abilities, specifically focusing on languages spoken in the Indian subcontinent. This set of benchmarks includes 4 different types of benchmarks. "IndicXNLI"[37] evaluates the impact of finetuning strategies, language models, multilinguality, and mixed-language input on pre-trained models. "IndicXParaphrase", a part of "Indic NLG"[38] investigates the performance of pre-trained language models on identifying paraphrased sentences, while analyzing the impact of different finetuning strategies, languages, and multilingual input. "IndicXCOPA"[39] tests pre-trained language models on their ability to answer cause-effect questions, while analyzing the impact of finetuning strategies, multilingualism, and language-specific nuances. "IndicXSentiment" evaluates the performance of pre-trained language models in classifying text sentiment, while analyzing the effects of finetuning strategies, multilinguality, and language-specific challenges.

Apart from this, human evaluation was also done by providing prompts to the models, and assessing the outputs.

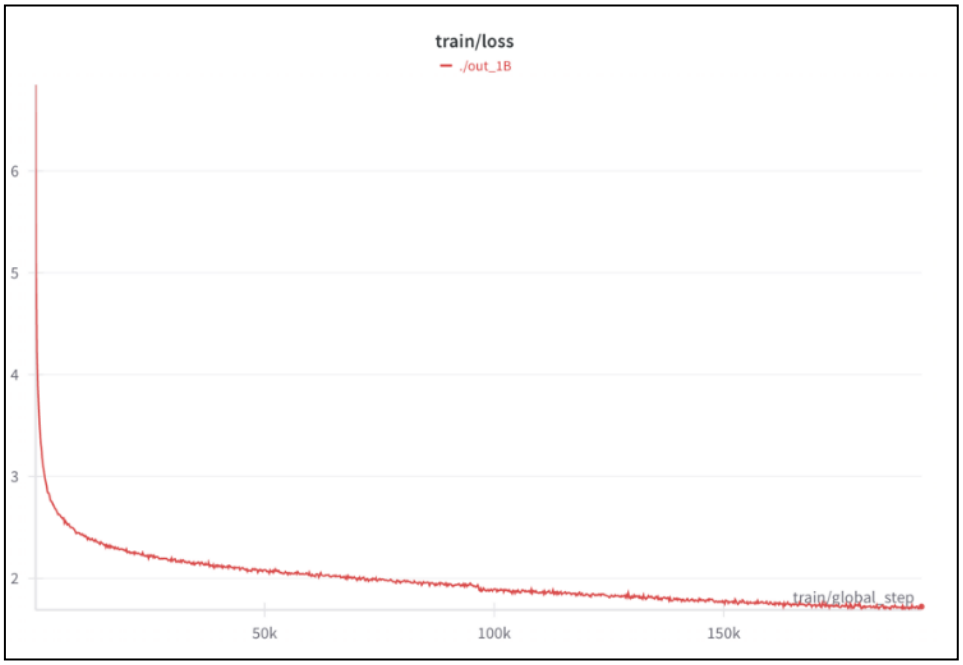

Figure 2. Loss curve for 1 Billion Parameter Model

V. RESULTS

This experiment resulted in two pretrained models, a 100M parameter model and one 1 billion parameter model. The main aim of the experiment was to study the performance of BitNet-1.58 on Hindi language. Nearly 2.5 times reduction in model size was noted, although the performance was not comparable to state-of-the-art models but given the size and number of parameters the performance was commendable. For testing the performance of model, perplexity was used and to study the performance on general NLP task, four tests were conducted, and accuracy was noted namely, IndicXNLI, IndicXParaphrase, IndicXCorpa and IndicXSentiment. **Table 3** shows the size of both the models.

This study developed two models, one with 100 million parameters and another with 1 billion parameters. The sizes of these models were 156MB and 2.1GB, respectively. In contrast, the original reference models had sizes of 450MB and 4.5GB. The reduction in model size could be due to factors such as optimizations in weight storage, different precision formats, or architectural modifications. Understanding these differences is important, as they may impact memory usage, computational efficiency, and overall model performance. Further investigation is needed to analyze how these variations affect real-world applications.

Table 3. Size of BitNet-1.58 Hindi Models

Model	Size	Size of comparable model
100M	156MB	450MB
1B	2.1GB	4.3GB

Perplexity was utilized in this paper to gauge the model's performance. One important statistic for assessing language models is perplexity, which shows how well a model predicts a sample of text. Perplexity basically measures how unsure a model is at guessing the subsequent word in a series. The more accurately the model predicts the text, the lower the perplexity. The perplexity of different models is assessed using a compressive set of prompts. Response of each of the models are recorded and perplexity of each response is calculated. Then the average perplexity is calculated. The results can be seen in **Table 4** and **Figure 3**.

The perplexity of AIRAVATA, a 7 billion Instruction Tuned Model was 6.831, it one of the most accurate models that exists for Hindi language. The perplexity score of this model is excellent. The perplexity score of the BitNet-1.58 100M and BitNet-1.58 1B was 42.6 and 12.8 respectively. The perplexity score is not state of the art but it proves the potential of the architecture. The following signifies that the model is able to remember the training dataset well. Training a larger model with over 3 billion parameter could achieve performance close to the full precision model.

During the evaluation of general NLP tasks, the model achieved an accuracy of approximately 50% across most test cases. The score of both the models are present in **Table 5**. This relatively low performance can largely be attributed to the absence of instruction tuning.

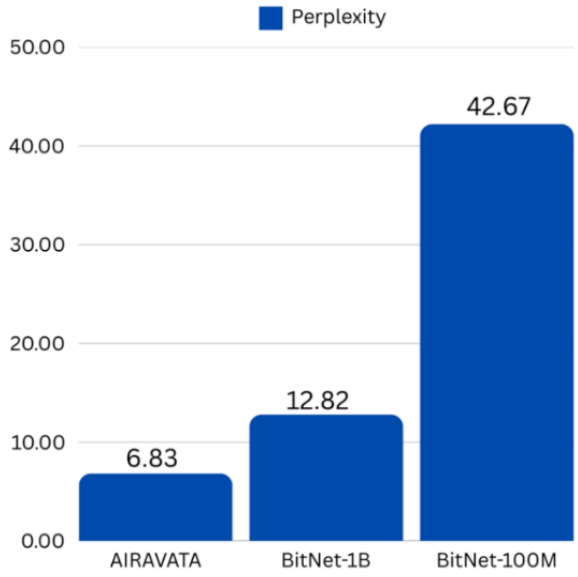

Figure 3. Perplexity

Many of the tasks in the testing framework required the model to follow specific instructions, yet without explicit fine-tuning for instruction-based responses, the model struggled to interpret and execute tasks effectively.

One major limitation was the reliance on generic pretraining, which, while useful for learning linguistic structures and semantic relationships, did not equip the model with the necessary adaptability for instruction-driven tasks. The lack of exposure to diverse task formats, structured prompts, and guided responses resulted in inconsistencies in output quality. Expanding the pretraining dataset to include a broader range of supervised and semi-supervised text corpora could help address this issue by enhancing the model's ability to generalize across multiple task types.

Table 3. Average Perplexity of Various Models

Model Name	Perplexity
AIRAVATA	6.831
BitNet-1.58 100M	42.677
BitNet-1.58 1B	12.823

Additionally, fine-tuning the model on carefully curated datasets for specific tasks can lead to improvements in both perplexity and task-based accuracy. Perplexity, a key metric for language model efficiency, measures how well a model predicts the next word in a sequence. A lower perplexity score typically indicates a better-trained model with a more refined understanding of language. By incorporating fine-tuning techniques such as instruction tuning—where the model is explicitly trained on instruction-response pairs—its ability to handle complex queries, reasoning tasks, and multi-step problem-solving can be significantly improved.

Table 4. Score of General NLP Task

Model	IndicXNLI	IndicXParaphrase	IndicXCopa	IndicXSentiment
100M	0.3281	0.5010	0.4944	0.5000
1B	0.3311	0.5147	0.5122	0.5481

Future work would focus on refining the training methodology by integrating instruction tuning, domain-specific fine-tuning, and exposure to real-world task variations. By systematically training the model on instructional prompts, command-based interactions, and structured datasets, its ability to understand and execute tasks can be greatly enhanced. This approach will not only improve performance on general NLP benchmarks but also ensure better adaptability for practical applications requiring high-level comprehension and contextual reasoning.

VI. Conclusion

By evaluating the effects of BitNet-1.58 architecture, this study investigates the effects of this architecture on model performance and size. The size of the model was reduced to nearly 2.5 times compared to original model. 150MB and 2.1GB were the size of model after converting the model to BitNet-1.58 which opens up new horizon and possibilities for the low bit models.

This study found that there was degradation in performance in the model, but the performance was still good. A perplexity of 42 and 12 were achieved for 100M and 1B parameter model respectively. Few problems that were noticed was the model tends to continuously repeat few sentences. This problem was mainly caused due to low amount of pre-training data, which reduced the knowledge space of the model and resulted in poor context interpretation and issue of repeated output. Also, in the testing for general NLP tasks, the model achieved an accuracy of about 50% for most tests, this is due to lack of instruction tuning as most of test relied on instructions given to the model. Using more pretraining data and finetuning the model for specific tasks could lead to better performance in both perplexity and general NLP task.

One major advantage of this method is reduction in size during inference. Reducing cost as well requiring less energy and computational resources. Looking on the other side of the coins, the biggest disadvantage being it requires to train a model from scratch. Taring of model using this architecture used the same resources compared to any traditional architecture. Other disadvantage is there is next to none improvement in inference time, in experiments conducted by this research, the inference time stayed on par with a full precision model.

VII. Future Scope

This research represents a crucial step toward advancing large-scale language models and expanding their capabilities in natural language processing (NLP). While a 1-billion-parameter model was successfully developed, there remains significant potential for scaling to even larger models. Due to resource constraints, training of models were limited to 150M and 1B parameters.

One of the key challenges identified was the model's suboptimal performance in handling general NLP tasks, particularly in Hindi. This issue cloud be solved using more training data and instruction tuning, which would increase the models knowledge base and improve context interpretation.

Another promising direction is the integration of feedback-driven learning techniques. Methods such as reinforcement learning with human feedback (RLHF) can help fine-tune model outputs, aligning them more closely with human expectations and improving response quality. Future research should explore hybrid approaches that combine supervised learning with unsupervised pretraining to create a balanced and efficient training framework.

VIII. Acknowledgment

The success of this research is the direct results of the support and contribution of various parties involved. A special thanks to Center of Artificial Intelligence in Medical Imaging and Forensics (CAIMIF), Sharda University for providing us with the required GPU and infrastructure for this experiment. Special thanks to Head of Department of Department of Computer Science and Engineering, Sharda University for allowing the use of resources and continuous support during this experiment. This experiment would not have been a success with them.

References

[1] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean, "Large Language Models in Machine Translation," Association for Computational Linguistics, 2007.

[2] H. Touvron *et al.*, "LLaMA: Open and Efficient Foundation Language Models," Feb. 2023, [Online]. Available: http://arxiv.org/abs/2302.13971

[3] OpenAI *et al.*, "GPT-4 Technical Report," Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.08774

[4] E. Almazrouei *et al.*, "The Falcon Series of Open Language Models," Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.16867

[5] B. Workshop *et al.*, "BLOOM: A 176B-Parameter Open-Access Multilingual Language Model," Nov. 2022, [Online]. Available: http://arxiv.org/abs/2211.05100

[6] Q. Jiang *et al.*, "Mistral 7B," Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.06825

[7] L. Team and A. @ Meta, "The Llama 3 Herd of Models," 2024.

[8] W. Wang *et al.*, "Model Compression and Efficient Inference for Large Language Models: A Survey," Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.09748

[9] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A White Paper on Neural Network Quantization," Jun. 2021, [Online]. Available: http://arxiv.org/abs/2106.08295

[10] J. Gala *et al.*, "Airavata: Introducing Hindi Instruction-tuned LLM," Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.15006
[11] S. Ma *et al.*, "The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits," Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.17764
[12] P. Dong *et al.*, "STBLLM: Breaking the 1-Bit Barrier with Structured Binary LLMs," Aug. 2024, [Online]. Available: http://arxiv.org/abs/2408.01803
[13] Y. Xu *et al.*, "OneBit: Towards Extremely Low-bit Large Language Models," Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.11295
[14] H. Wang *et al.*, "BitNet: Scaling 1-bit Transformers for Large Language Models," Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.11453
[15] J. Wang *et al.*, "1-bit AI Infra: Part 1.1, Fast and Lossless BitNet b1.58 Inference on CPUs," Oct. 2024, [Online]. Available: http://arxiv.org/abs/2410.16144
[16] H. Wang, S. Ma, and F. Wei, "BitNet a4.8: 4-bit Activations for 1-bit LLMs," Nov. 2024, [Online]. Available: http://arxiv.org/abs/2411.04965
[17] J. Wang *et al.*, "Bitnet.cpp: Efficient Edge Inference for Ternary LLMs," Feb. 2025, [Online]. Available: http://arxiv.org/abs/2502.11880
[18] J. Chee, Y. Cai, V. Kuleshov, and C. De Sa, "QuIP: 2-Bit Quantization of Large Language Models With Guarantees," Jul. 2023, [Online]. Available: http://arxiv.org/abs/2307.13304
[19] J. Li *et al.*, "Fast and Efficient 2-bit LLM Inference on GPU: 2/4/16-bit in a Weight Matrix with Asynchronous Dequantization," Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.16442
[20] Y. Liu *et al.*, "VPTQ: Extreme Low-bit Vector Post-Training Quantization for Large Language Models," Sep. 2024, [Online]. Available: http://arxiv.org/abs/2409.17066
[21] Z. Liu *et al.*, "ParetoQ: Scaling Laws in Extremely Low-bit LLM Quantization," Feb. 2025, [Online]. Available: http://arxiv.org/abs/2502.02631
[22] Y. Zhao *et al.*, "Atom: Low-bit Quantization for Efficient and Accurate LLM Serving," Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.19102
[23] Y. Chen *et al.*, "BitMoD: Bit-serial Mixture-of-Datatype LLM Acceleration," Nov. 2024, [Online]. Available: http://arxiv.org/abs/2411.11745
[24] W. Huang *et al.*, "BiLLM: Pushing the Limit of Post-Training Quantization for LLMs," Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.04291
[25] S. Kim *et al.*, "SqueezeLLM: Dense-and-Sparse Quantization," Jun. 2023, [Online]. Available: http://arxiv.org/abs/2306.07629
[26] Y. Li and P. Panda, "TesseraQ: Ultra Low-Bit LLM Post-Training Quantization with Block Reconstruction," Oct. 2024, [Online]. Available: http://arxiv.org/abs/2410.19103
[27] Y. Shang, Z. Yuan, Q. Wu, and Z. Dong, "PB-LLM: Partially Binarized Large Language Models," Sep. 2023, [Online]. Available: http://arxiv.org/abs/2310.00034
[28] J. Lin *et al.*, "AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration," Jun. 2023, [Online]. Available: http://arxiv.org/abs/2306.00978
[29] Y. Guo *et al.*, "PT-BitNet: Scaling up the 1-Bit Large Language Model with Post-Training Quantization." [Online]. Available: https://ssrn.com/abstract=4987078
[30] S. Liu, Z. Liu, X. Huang, P. Dong, and K.-T. Cheng, "LLM-FP4: 4-Bit Floating-Point Quantized Transformers," Oct. 2023, doi: 10.18653/v1/2023.emnlp-main.39.
[31] D. Du *et al.*, "BitDistiller: Unleashing the Potential of Sub-4-Bit LLMs via Self-Distillation," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2024, pp. 102–116. doi: 10.18653/v1/2024.acl-long.7.
[32] Z. Liu *et al.*, "LLM-QAT: Data-Free Quantization Aware Training for Large Language Models," May 2023, [Online]. Available: http://arxiv.org/abs/2305.17888
[33] H. Xia *et al.*, "FP6-LLM: Efficiently Serving Large Language Models Through FP6-Centric Algorithm-System Co-Design," Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.14112
[34] H. Chen *et al.*, "DB-LLM: Accurate Dual-Binarization for Efficient LLMs," Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.11960
[35] Raghavan, M. Amer, G. Taylor, and S. Chai, "BIT-REGULARIZED OPTIMIZATION OF NEURAL NETS."
[36] S. Parida, S. Panwar, K. Lata, S. Mishra, and S. Sekhar, "Building pre-train LLM Dataset for the INDIC Languages: a case study on Hindi," Jul. 2024, [Online]. Available: http://arxiv.org/abs/2407.09855
[37] D. Aggarwal, V. Gupta, and A. Kunchukuttan, "INDICXNLI: Evaluating Multilingual Inference for Indian Languages." [Online]. Available: https://indicxnli.github.
[38] Kumar *et al.*, "IndicNLG Benchmark: Multilingual Datasets for Diverse NLG Tasks in Indic Languages." [Online]. Available: https://en.wikipedia.org/wiki/
[39] M. S. U. R. Khan *et al.*, "IndicLLMSuite: A Blueprint for Creating Pre-training and Fine-Tuning Datasets for Indian Languages," Mar. 2024, [Online]. Available: http://arxiv.org/abs/2403.06350