Go Small or Go Home:
Exploring lightweight generative models for mobile devices

Anvesha Katariyar* Aryaman Shandilya* Eman Ansar*
{aakatari, aryamans, eansar}@andrew.cmu.edu

1 Introduction

Motivation The growing demand for on-device AI necessitates efficient generative models for text and image generation on mobile devices. Traditional models, despite their effectiveness, often have high computational and memory costs, making them unsuitable for mobile deployment. This project addresses these challenges by optimizing lightweight, mobile-compatible generative models. While advancements like TinyBERT exist, further optimization is needed to reduce latency, minimize memory usage, and improve responsiveness. Enhanced mobile-friendly models can unlock new applications in social media, real-time language assistance, and AR/VR.

Objective This project aims to adapt and implement lightweight generative models for mobile devices, focusing on advanced efficiency techniques such as quantization, pruning, and knowledge distillation. By reducing memory usage, accelerating inference, and maintaining output quality, we aim to create highly efficient models suited for the constraints of mobile hardware.

2 Datasets and Tasks

2.1 Datasets

2.1.1 Text

- Dataset: WikiText-103
- Description: A large-scale text dataset derived from English Wikipedia articles, with a vocabulary of about 267k words. This dataset is ideal for training language models that generate coherent and grammatically correct text.
- Usecase: Since our project focuses on lightweight generative models, WikiText-103 can be used for tasks like text completion.

* Authors contributed equally

2.1.2 Image

- Dataset: Subset of ImageNet
- Description: A large dataset with millions of labeled images across thousands of categories. While the full ImageNet dataset would be computationally intensive, we can use a smaller subset for training compact models.
- Usecase: To create high-quality images with more diversity, we can use a subset of ImageNet for high-resolution image generation on mobile devices.

2.2 Tasks, Metrics for Evaluation

2.2.1 Performance on Text Data

- BLEU Score: It measures the precision of overlapping n-grams between the candidate and reference texts while applying a brevity penalty to discourage overly short outputs. Ranging from 0 to 1, where 1 indicates a perfect match, BLEU focuses on exact matches of words and phrases, making it effective for corpus-level evaluation.

2.2.2 Performance on Image Data

- Inception Score (IS): This metric evaluates the quality of generated images by analyzing their diversity and the confidence of a pre-trained classifier. A higher score means the model generates high-quality images with more diversity.
- Fréchet Inception Distance (FID): FID compares the distribution of generated images to real images and measures the similarity between these distributions. Lower FID scores indicate better image quality.

2.2.3 Efficiency

- Inference Time: We will measure how long it takes for the model to generate output, as real-time performance is crucial for practical applications especially on mobile devices.

- Model Size: We will track the model's size in terms of the number of parameters and storage space. Smaller models will be more suitable for deployment on mobile devices.
- Memory Usage: We will track how much memory (RAM) is consumed by the model during inference. Efficient models should use less memory to reduce the chances of memory overflows or app crashes.

3 Related Work

The growing demand for deploying deep learning models on resource-constrained devices has spurred advancements in lightweight generative models. Techniques such as knowledge distillation, quantization, and pruning have been pivotal in optimizing these models. Hinton et al. (2015) (Hinton et al., 2015) introduced knowledge distillation, enabling smaller "student" models to achieve performance comparable to larger "teacher" models while reducing computational overhead. Building on this, Aguinaldo et al. (2019) applied distillation to GANs, compressing model sizes without compromising output quality (Aguinaldo et al., 2019), while Jiao et al. (2020) demonstrated similar success with TinyBERT in NLP tasks (Jiao et al., 2020). Zhou et al. (2024) (Zhou et al., 2024) further extended these methods through dual-faceted knowledge distillation, optimizing both performance and efficiency for lightweight models. These works underscore the versatility of knowledge distillation in achieving effective model compression across domains.

For image-to-image translation, Tang et al. (2022) introduced Region-Aware Knowledge Distillation (ReKo), which uses attention modules and patch-wise contrastive learning to identify and transfer critical image features (Tang et al., 2022). ReKo demonstrated that compressed CycleGAN models could outperform their larger teachers in tasks like Horse-to-Zebra translation, showcasing the effectiveness of region-specific distillation for generative models.

Han et al. (2016) pioneered Deep Compression by combining pruning, quantization, and Huffman coding to significantly reduce model size and computational complexity with minimal accuracy loss (Han et al., 2016). Building on this, Chen et al. (2020) applied pruning to GANs, achieving mobile-optimized generative models that retained high-quality outputs (Chen et al., 2020).

Li et al. (2023) introduced DreamTeacher, a framework for pretraining image backbones using generative models, which achieved state-of-the-art performance in dense prediction tasks (Li et al., 2023). DreamTeacher highlighted the potential of generative models as pretraining frameworks for improved efficiency and downstream task performance, offering valuable insights for optimizing DCGAN architectures.

For GANs, methods like ReKo's attention-driven distillation and DreamTeacher's integration of generative backbones emphasize maintaining output quality while minimizing size and latency. Similarly, TinyBERT's success with distillation and pruning provides a blueprint for extending these techniques to generative models, paving the way for efficient, resource-friendly solutions across both vision and NLP applications.

4 Approach

4.1 Baseline Models

4.1.1 Llama-2-7B (Text Generation)

Llama-2-7B will serve as the baseline for text generation tasks. It is one of the latest transformer-based language models, and we will use it to compare against more advanced optimizations. Key characteristics include reduced model size and optimized transformer layers.

4.1.2 Deep Convolutional GAN (DCGAN) (Image Generation)

Deep Convolutional GAN (DCGAN) will serve as the baseline for image generation tasks due to its simplicity, stability, and wide adoption in GAN research. DCGAN replaces fully connected layers with convolutional ones, uses transposed convolutions for upsampling, and incorporates batch normalization and Leaky ReLU activations to improve convergence and gradient flow. These design choices make it a reliable reference for evaluating advancements in GAN optimization

4.2 Proposed Methods

The key goal of the project is to improve the efficiency and performance of Llama-2 and DCGAN for on-device use by employing various optimization techniques. Here's a breakdown of the methods we will use:

4.2.1 Knowledge Distillation

Llama-2 Train a smaller "student" version of Llama-2 using knowledge distillation from a larger,

pre-trained model. This process involves transferring the knowledge (soft labels) from a larger model to the smaller model, preserving accuracy while reducing size.

DCGAN Apply a similar distillation technique to DCGAN, where a larger GAN serves as the teacher. The aim is to make DCGAN more efficient while retaining the quality of generated images by distilling its knowledge into a smaller, more mobile-friendly architecture.

4.2.2 Quantization and Pruning

Quantization Implement both static and dynamic quantization for Llama-2 and DCGAN. This involves reducing the precision of the model's weights (e.g., from 32-bit floating-point to 8-bit integers) to reduce memory usage and speed up inference time, without significantly impacting performance. We will experiment with different levels of precision to find the balance that works best for mobile devices.

Pruning Use structured pruning techniques, such as removing entire attention heads or convolutional layers, to reduce the number of parameters in both Llama-2 and DCGAN. For example, for Llama-2, we can prune attention heads that contribute minimally to the model's output, and for DCGAN, pruning some of the convolutional layers could reduce the computational cost without significantly affecting the image quality.

4.3 Performance Evaluation Metrics

4.3.1 Text Generation

Evaluate Llama-2 and our model using traditional NLP metrics such as the BLEU Score. These metrics will help assess the fluency and coherence of generated text. Additionally, inference latency, model size, and memory usage will be tracked as part of the mobile optimization evaluation.

4.3.2 Image Generation

Evaluate DCGAN using Fréchet Inception Distance (FID), which compares the generated images to real images, and Inception Score (IS) to measure diversity and quality. Like Llama-2, performance will also be measured in terms of latency, model size, and memory consumption.

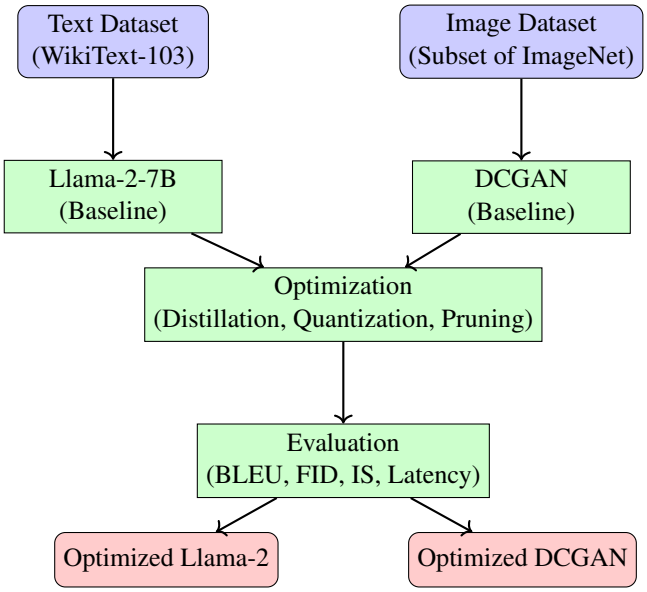

Figure 1: Workflow for optimizing lightweight generative models for mobile devices.

5 Experiments

5.1 Experimental Setup

Hardware All experiments will be run on NVIDIA GPUs (A100) for both training and inference tasks. For mobile deployment, we will evaluate latency on an Android device (e.g., Pixel 6).

Implementation Details We implement our models using PyTorch and leverage HuggingFace for Llama-2. For DCGAN, we use the provided implementation with modifications for quantization and pruning. All models are trained with Adam optimizers.

5.2 Results

We present the results of our baseline models in Tables 1 and 2. These tables summarize the metrics we plan to evaluate for text generation using TinyBERT and image generation using Llama-2, respectively. Currently, the tables serve as placeholders, and detailed results will be added as experiments progress.

Text Generation Results (Table 1) Table 1 outlines the evaluation metrics for TinyBERT and its optimized versions. The metrics include:
BLEU Score: This measures the quality and coherence of text generated across multiple NLP tasks by measuring n-gram overlap.
Latency (ms): This indicates the time required for

inference, which is crucial for deployment on mobile devices.
Model Size: This reflects the memory efficiency of the model after optimization techniques such as quantization and pruning.

By comparing the baseline Llama-2 with its quantized and pruned versions, we aim to understand the trade-offs between computational efficiency and task performance.

Image Generation Results (Table 2) Table 2 presents baseline results for DCGAN along with placeholders for its optimized versions. The evaluation metrics include: **FID Score (Fréchet Inception Distance)**: This measures the realism of generated images by comparing them to real images. **IS Score (Inception Score)**: This evaluates the diversity and quality of generated images. **Model Size (MB)**: This metric indicates the memory efficiency after applying optimization techniques.

Visual Representation of Results To complement the tables, we plan to include line plots and bar charts that visualize the impact of optimization techniques. For instance:

- A line plot showing the relationship between model size and inference latency for both Llama-2 and DCGAN will highlight the trade-offs of different optimization strategies.
- Bar charts comparing the BLEU score and FID score across baseline and optimized models will provide an intuitive comparison of task performance and image quality.

These visualizations will be integrated into the final report to support our findings and provide a clearer understanding of the results.

Model	BLEU Score	Latency	Model Size(GB)
Llama-2-7B	0.32	96s	13.5
Llama-2-7B (Quantization)	-	-	-
Llama-2-7B (Pruning)	-	-	-
Llama-2-7B (Both)	-	-	-

Table 1: Results for text generation.

Model	FID Score	IS Score	Model Size (MB)
DCGAN	238	6.82	23.69
DCGAN (Quantization)	-	-	-
DCGAN (Pruning)	-	-	-
DCGAN (Both)	-	-	-

Table 2: Results for image generation.

6 Plan

To ensure effective collaboration and equal contribution, the tasks will be divided among the three team members as follows:

6.1 Optimization of Llama-2 (Aryaman Shandilya and Eman Ansar)

- **Knowledge Distillation**:Implement teacher-student training to create a smaller version of Llama-2. Test various configurations to balance size reduction and quality retention.
- **Quantization**: Apply static and dynamic quantization methods to reduce precision and improve efficiency.
- **Evaluation**: Use BLEU Score to measure the model's text generation performance. Measure latency, model size, and memory usage to ensure mobile compatibility.

6.2 Optimization of DCGAN (Eman Ansar and Anvesha Katariyar)

- **Knowledge Distillation**:Train a smaller GAN model using a teacher-student approach.Focus on maintaining image diversity and quality during distillation.
- **Pruning**:Implement structured pruning to remove less impactful layers (e.g., convolutional layers). Experiment with different pruning thresholds to minimize performance loss.
- **Evaluation**: Use FID and Inception Score (IS) to measure image quality and diversity. Measure latency and memory usage for real-time image generation.

6.3 Integration and Comparison (All Team Members)

- Compare optimized Llama-2 and DCGAN models against their respective baselines.
- Conduct final benchmarking for efficiency and quality.

6.4 Pair Programming and Collaboration

- **Knowledge Distillation**:
  - Aryaman & Eman: Llama-2
  - Eman & Anvesha: DCGAN
- **Evaluation Metrics**:
  - Eman & Anvesha will implement metrics for consistency.
  - GitHub for version control, regular commits to avoid conflicts.
  - Weekly progress meetings; additional sessions as needed.

References

Angeline Aguinaldo, Ping-Yeh Chiang, Alex Gain, Ameya Patil, Kolten Pearson, and Soheil Feizi. 2019. Compressing gans using knowledge distillation.

Hanting Chen, Yunhe Wang, Han Shu, Changyuan Wen, Chunjing Xu, Boxin Shi, Chao Xu, and Chang Xu. 2020. Distilling portable generative adversarial networks for image translation.

Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding.

Daiqing Li, Huan Ling, Amlan Kar, David Acuna, Seung Wook Kim, Karsten Kreis, Antonio Torralba, and Sanja Fidler. 2023. Dreamteacher: Pretraining image backbones with deep generative models.

Minghan Tang, Ajay Divakaran, and Bryan A. Plummer. 2022. Image into reasoning: Text-supervised causal scene inference.

Bojun Zhou, Tianyu Cheng, Jiahao Zhao, Chunkai Yan, Ling Jiang, Xinsong Zhang, and Juping Gu. 2024. Enhancing few-shot learning in lightweight models via dual-faceted knowledge distillation.