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

- GLUE Benchmark: It consists of multiple tasks designed to test models on linguistic skills such as sentiment analysis, sentence similarity, and logical inference. The benchmark provides an overall score, enabling a standardized comparison of model performance across diverse language tasks.

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 rising demand for deep learning on resource-limited devices has driven research into lightweight generative models, with knowledge distillation, introduced by Hinton et al.(2015),.(Hinton et al., 2015) as a key approach. This technique transfers knowledge from a large "teacher" model to a smaller "student" model, achieving comparable performance with reduced computational overhead. Building on this, Han et al. (2016) (Han et al., 2016) proposed Deep Compression, which combines pruning, quantization, and Huffman coding to compress deep neural networks, enabling efficient deployment without substantial accuracy loss.

For natural language processing, Jiao et al. (2020)(Jiao et al., 2020) demonstrated the effectiveness of distillation through TinyBERT, which compresses the BERT model while maintaining performance on natural language understanding tasks. This highlights the potential of distillation for high-dimensional data representation, a critical aspect for lightweight generative models.

In the context of generative adversarial networks (GANs), Aguinaldo et al. (2019)(Aguinaldo et al., 2019) adapted knowledge distillation to compress GANs, focusing on reducing their size while retaining their ability to generate high-quality outputs. Chen et al. (2020)(Chen et al., 2020) extended this idea by introducing a distillation framework tailored for image translation tasks. Their method produces portable GANs optimized for mobile devices, striking a balance between model size and image quality.

4 Approach

4.1 Baseline Models

4.1.1 TinyBERT (Text Generation)

TinyBERT will serve as the baseline for text generation tasks. It is already a lightweight transformer-based model designed for mobile environments, and we will use it to compare against more advanced optimizations. Key characteristics include reduced model size (compared to BERT) 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 TinyBERT 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

TinyBERT Train a smaller "student" version of TinyBERT using knowledge distillation from a larger, pre-trained BERT 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 TinyBERT 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 TinyBERT and DCGAN. For example, for TinyBERT, 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 TinyBERT and our model using traditional NLP metrics such as the GLUE Benchmark. These metrics will help assess the fluency and coherence of generated text. Additionally, latency (in milliseconds), 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 TinyBERT, performance will also be measured in terms of latency, model size, and memory consumption.

5 Expected Outcomes

Model Size Reduction Expect to achieve significant reductions in both TinyBERT's and DCGAN's model sizes while retaining performance. TinyBERT might be reduced from a few hundred MBs to just a few tens of MBs, while DCGAN could see similar reductions.

Improved Inference Speed You'll observe faster inference times on mobile devices due to the application of quantization, pruning, and other architectural changes. In some cases, you could aim for reductions in inference time by up to 50% or more compared to the baseline.

Quality Retention Despite optimizations, the generated text and images should maintain acceptable levels of quality. For TinyBERT, we may see minor trade-offs in text coherence for significant efficiency gains, and for DCGAN, image quality should stay high with minor degradation.

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 TinyBERT (Aryaman Shandilya and Eman Ansar)

- **Knowledge Distillation**:Implement teacher-student training to create a smaller version of TinyBERT. 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 GLUE Benchmark 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 TinyBERT and DCGAN models against their respective baselines.
- Conduct final benchmarking for efficiency and quality.

6.4 Pair Programming and Collaboration

- **Knowledge Distillation**:
  - Aryaman & Eman: TinyBERT
  - 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.